VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA ŠIFROVACÍCH ALGORITMŮ VE STANDARDU 802.11

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

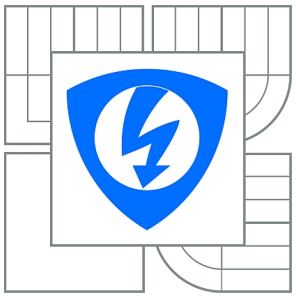
AUTOR PRÁCE
AUTHOR

Bc. JINDŘICH VOJTÍŠEK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA ŠIFROVACÍCH ALGORITMŮ VE STANDARDU 802.11

ANALYSIS OF CRYPTOGRAPHIC ALGORITHMS 802.11

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

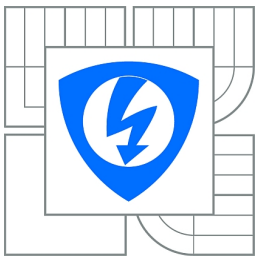
AUTOR PRÁCE — Bc. JINDŘICH VOJTÍŠEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. BOHUMIL NOVOTNÝ

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Jindřich Vojtíšek ***ID:*** 120811
Ročník: 2 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Analýza šifrovacích algoritmů ve standardu 802.11

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce vypracujte teoretický rozbor jednotlivých typů zabezpečovacích metod a jejich slabin. Seznamte se s používanými šifrovacími algoritmy ve standardu 802.11, především WPA a WPA2. Vytvořte model klíčování v bezdrátových sítích s důrazem na bezpečnost sítě. Model vytvořte v programu Matlab. Model musí obsahovat grafický výstup průběhu zabezpečení a následného šifrování komunikace v závislosti na modelovaném typu zabezpečení.

DOPORUČENÁ LITERATURA:

[1] Liu, D. Q., Coslow, M.: Extensible authentication protocols for IEEE standards 802.11 and 802.16. In Proceedings of the International Conference on Mobile Technology, Applications, and Systems, Mobility '08, New York, NY, USA: ACM, 2008, ISBN 978-1-60558-089-0

[2] J. Salowey, L. Dondeti, V.Narayanan and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)," RFC 5295, Internet Eng. Task Force, 2008

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 30.5.2014

Vedoucí práce: Ing. Bohumil Novotný
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá standardem 802.11 a hlavně algoritmy pro zabezpečení dat v nich. Je zde proveden rozbor algoritmů WEP, WPA a WPA2, kdy u těchto algoritmů je zde popsáno, jakým způsobem probíhá šifrování pomocí jednotlivých algoritmů. K tomuto účelu je práce doplněna o bloková schémata těchto metod.
V praktické části je provedena realizace šifrovacích algoritmů WEP, WPA a WPA2 v prostředí Matlab simulink. Modely jsou doplněny grafickými výstupy pro pochopení, jak se mění data během průchodu tímto systémem.

KLÍČOVÁ SLOVA

802.11, AES, CCMP, MIC, RC4, TKIP, WEP, WPA, WPA2

ABSTRACT

This work deals with wireless standard 802.11, primaly about security algorithms used in them. Further there is made analysis of algorithms WEP, WPA and WPA2. This algorithms are described how coding by them works and for easier understandig are added block schemes of their principles.
In practical part is realized algorithms WEP, WPA and WPA2 in program Matlab simulink. Model is complemented by graphs which shows how data changes when comming throught this systems.

KEYWORDS

802.11, AES, CCMP, MIC, RC4, TKIP, WEP, WPA, WPA2

VOJTÍŠEK, Jindřich *Analýza šifrovacích algoritmů ve standardu 802.11*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013/2014. 89 s. Vedoucí práce byl Ing. Bohumil Novotný,

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Analýza šifrovacích algoritmů ve standardu 802.11“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Bohumilu Novotnému za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Standard 802.11, známý také jako Wi-Fi, se díky svým vlastnostem, jako jsou nepotřeba kabeláže, mobilita, využití nelicencovaných pásem a cena, stal velice populárním. Z principů šíření radiového signálu vyplývá, že tuto komunikaci lze snadno odposlouchávat, což je problém v situaci, kdy je potřeba přenášet citlivé informace. Dále zde může nastat problém s neoprávěným využíváním nechráněné sítě nebo může dojít i ke krádeži identity. Proto byly vytvořeny šifrovací algoritmy WEP, WPA a WPA2, které mají za cíl tyto informace chránit a mají zabránit lidem ve zneužívání přístupu do sítě, do které nemají povolený přístup.

Tato práce má za cíl seznámit s principy, které jsou použity v jednotlivých šifrovacích algoritmech. Principy jednotlivých typů zabezpečení budou vysvětleny v teoretické části. Toto bude doplněno o vhodné obrázky pro jednodušší pochopení průběhu zabezpečení. Pro pochopení funkce některých bloků budou doplněny i pseudokódy, které ukazují principy fungování jednotlivých bloků a poskytují snadnější orientaci v operacích prováděných uvnitř bloků. Bude proveden rozbor slabých míst jednotlivých algoritmů s popisem problému a jak je možné tuto chybu zneužít.

V praktickém řešení budou realizovány jednotlivé šifrovací algoritmy. Zvolené prostředí pro tvorbu těchto modelů je Matlab simulink. Jelikož zvolený program Matlab umožňuje vytvářet funkční blokové modely s praktickými grafickými výstupy, je možné snadněji ukázat principy šifrovacích algoritmů a napomoci tak jejich pochopení. Realizace bude obsahovat zdrojové kódy bloků, které bude nutné pro potřeby modelování jednotlivých algoritmů vytvořit.

1 STANDARD IEEE 802.11

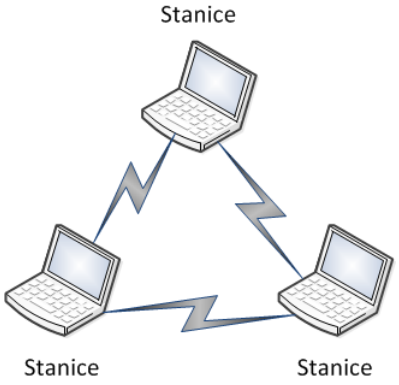
U pevných sítí platí, že adresa reprezentuje umístění stanice, naopak u bezdrátových sítí je tato adresa nezávislá na přesné poloze stanice v rámci sítě, tedy není známa přesná poloha stanice. Z tohoto nám vyplývá, že v rámci komunikace v bezdrátové síti mezi dvěmi zařízeními je nutné signál vysílat v rámci celého okolního prostředí. To v praxi znamená, že tuto komunikaci může zachytit kterákoliv stanice v rámci okolí, i když jí data nejsou určena, protože v rádiovém prostředí se signál šíří všesměrově. Proto je bezdrátová komunikace označována jako přenos po sdíleném médiu.

IEEE 802.11 vyžaduje, aby se bezdrátová sít jevila vyšším vrstvám jako sít LAN. Tohoto je dosaženo tím, že vrstvy pod vrstvou MAC musí implementovat algoritmy pro bezdrátové sítě, například řešení mobility stanice. Podrobnější informace k Wi-Fi na [13], [16] a [18].

1.1 Struktura bezdrátové sítě

1.1.1 Ad-hoc (IBSS)

V této síti je několik stanic, které mohou libovolně mezi sebou komunikovat přímo. Z tohoto vyplývá, že se zde netvoří žádná hierarchická struktura. Pro některé případy v praxi je toto řešení vhodné, ale ve většině případů se nepoužívá.



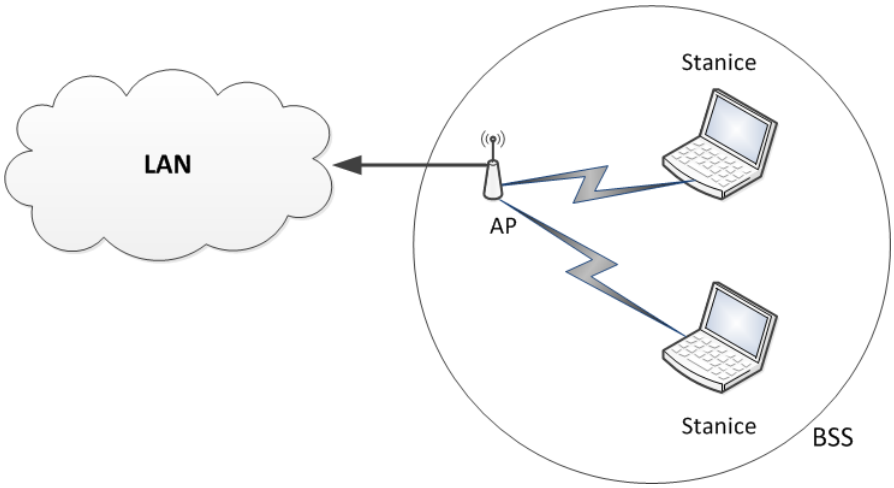
Obr. 1.1: Ad-hoc struktura bezdrátové sítě

1.1.2 Infrastrukturní síť

Jedná se o nejobvyklejší řešení bezdrátových sítí v praxi. Základní částí je takzvaný BSS (Basic Service Set), což je oblast, kde se kromě vlastních stanic vyskytuje i pří-

stupový bod (dále AP). Narozdíl od stanic má AP schopnost komunikovat s jinými typy sítí, například LAN. K tomu, abychom byli schopni rozeznat jednotlivé AP, je zde zaveden parametr BSSID (Basic Service Set Identifier), který je unikátní pro každý přístupový bod. BSSID má stejnou strukturu jako MAC adresa v pevných sítích a je tedy součástí MAC hlavičky při komunikaci mezi AP a stanicí. V této síti už nemohou stanice komunikovat přímo mezi sebou, ale musí směrovat svou komunikaci přes AP.

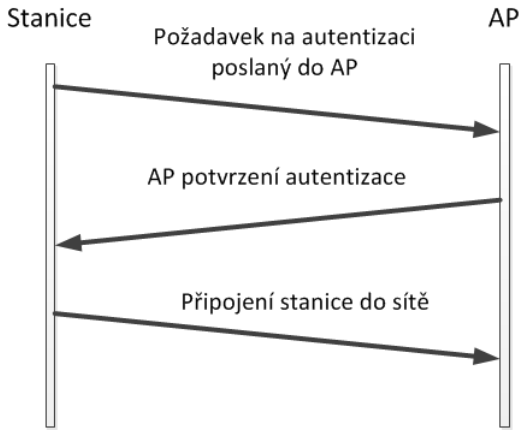
Kromě BSSID je používáno jestě SSID (Service Set Identifier) o délce 0 až 32 bajtů. SSID poskytuje textové označení sítě, tak aby bylo jméno sítě čitelné pro člověka.



Obr. 1.2: Infrastrukturní bezdrátová síť

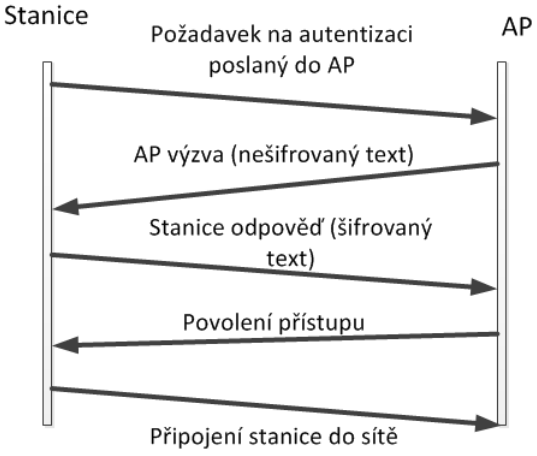
1.2 Autentizace v IEEE 802.11

- Otevřená autentizace, kdy AP přijme každé zařízení bez ověření jejich identity.



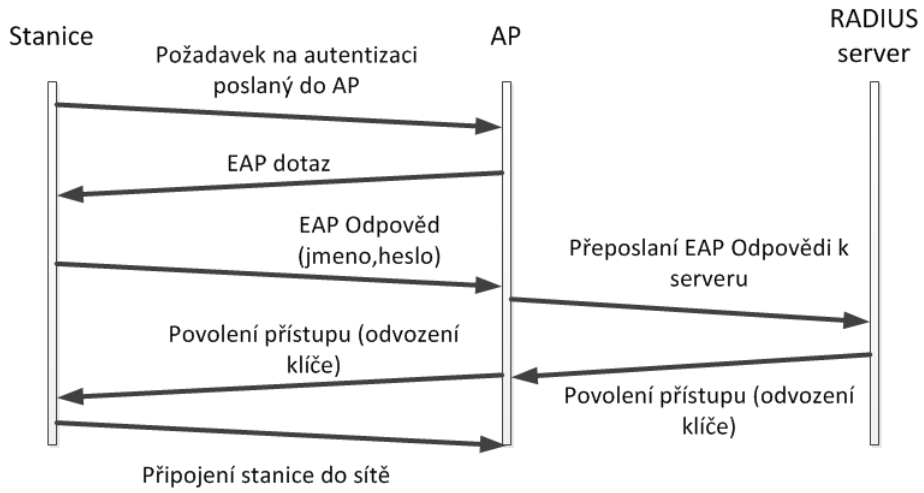
Obr. 1.3: Autentizace v otevřeném systému

- Autentizace sdíleným klíčem, kdy AP přijme každé zařízení, které zná tajný klíč. Tato autentizace probíhá v několika krocích. Nejprve stanice vyšle požadavek pro připojení do sítě na AP, kdy v reakci na tento požadavek AP odešle stanici odpověd obsahující text, který AP vyžaduje zašifrovat pro ověření identity stanice. Stanice provede podle svého klíče šifrování textu a odešle zpět k AP. AP dešifruje text z paketu a dle výsledku rozhodne o stanici, jestli má právo se připojit do sítě a rozhodnutí odešle zpět ke stanici.



Obr. 1.4: Autentizace sdíleným klíčem

- 802.1X, poskytuje nejsilnější zabezpečení za cenu dalšího zařízení v síti, autentizačního serveru. Nejčastější řešení využívá protokol EAP a RADIUS autentizační server. Aby stanice mohla být registrovaná, musí poslat na AP uživatelské jméno a heslo nebo digitální certifikát. AP tyto informace přepošle do RADIUS serveru, který ověří, jestli tento záznam má ve svých tabulkách a rozhodne, jestli má uživatel právo se připojit. Toto řešení je nejčastěji používané ve velkých bezdrátových sítích v kombinaci s WPA nebo WPA2 šifrováním. Pro WPA a WPA2 platí, že ve fázi povolení přístupu se provede odvození klíčů pro vlastní WPA šifrování. Detailní informace k různým druhům zabezpečení jsou k dispozici na [2].



Obr. 1.5: Autentizace dle 802.1X

1.3 Vývoj standardů v rámci IEEE 802.11

1.3.1 IEEE 802.11

Jedná se o první standard vydaný v roce 1997 společností Wi-Fi Alliance. Využívá modulaci DSSS a FHSS a pracuje v nelicencovaném frekvenčním pásmu 2,4 GHz, maximální rychlost je 2 Mb/s. Jelikož nebyly standardizovány jednotlivé kanály, docházelo k nekompatibilitě zařízení mezi různými výrobci.

DSSS(přímé rozprostírání signálu)

Datový tok je pomocí XOR sloučen s pseudonáhodnou posloupností získanou dle Barkerovy sekvence a následně zakódován pomocí BPSK či QPSK.

FHSS (frekvenční přeskoky)

Je metoda sloužící k rychlému přepínání mezi jednotlivými kanály, k tomu se využívá pseudonáhodné posloupnosti, která je známá vysílači i přijímači. Toto nám umožňuje sdílet frekvenční pásmo s dalšími typy přenosů s minimálními interferencemi.

1.3.2 IEEE 802.11a

Tento standard vyšel v roce 1999. Oproti původnímu standardu se zde změnilo frekvenční pásmo do nelicencovaného pásma na 5 GHz a původní modulace DSSS byla nahrazena OFDM. Podporované rychlosti tohoto standardu jsou 6, 12, 24, 36, 48 a 52 Mb/s.

OFDM

OFDM znamená v češtině ortogonální multiplex s kmitočtovým dělením. Každý z původních 12 kanálů je rozdělen na 52 subkanálů (48 datových a 4 pilotních) a každý z nich je klíčován vícestavovými modulacemi BPSK, QPSK, 16-QAM nebo 64-QAM. Ortogonalita pak zajišťuje, že subnosné jsou k sobě vzájemně ortogonální a tedy jejich skalární součin je roven nule.

Výhodou OFDM je odolnost proti rušení mnohacestného šíření signálu, proti interferenci mezi symboly a interferenci mezi nosnými frekvencemi.

1.3.3 IEEE 802.11b

Tento vyšel ve stejnou dobu jako IEEE 802.11a a jedná se o vylepšení standardu IEEE 802.11. Kromě modulace DSSS přidává jestě CKK. Došlo k standardizaci přenosových kanálů na 13 kanálů rozložených na frekvencích od 2,412 GHz do 2,472 GHz a k navýšení rychlosti na 11 Mb/s.

CKK (doplňkové kódové klíčování)

Jedná se o skupinu 64 kódových slov o délce 8 bitů. Kódová slova mají unikátní matematickou vlastnost, která umožňuje tyto kódy rozeznat i při zarušení signálu.

V rámci Wi-Fi to znamená že na začátek rámce se místo 1 bitu nyní dává bitová sekvence o 8 b, která se využije k rozprostření signálu. Tento kód poskytuje dobrou odolnost proti rušení z mnohacestného šíření signálu.

1.3.4 IEEE 802.11e

V roce 2005 došlo k důležité úpravě a to zavedení kvality služeb (QoS) a prioritizace. To umožnilo, aby ve Wi-Fi síti bylo efektivně možné požívat služby závislé na zpoždění, například přenos hlasu.

1.3.5 IEEE 802.11g

V roce 2003 vyšel tento standard, který měl za cíl zvýšit propustnost bezdrátové sítě, výsledkem bylo navýšení přenosové rychlosti až na 54 Mb/s. Toho se dosahuje kombinováním vlastností ze standardů IEEE 802.11a a IEEE 802.11b. Používá frekvenční pásmo na 2,4 GHz, ale jako modulace se používá DSSS a OFDM. Pro rychlosti do 11 Mb/s se používá DSSS a pro vyšší se pak přejde k použití OFDM.

Výhodou tohoto standardu je i to, že je kompaktibilní s IEEE 802.11b avšak pouze do rychlosti 11 Mb/s. Proto v praxi bývá tento standard označován často jako IEEE 802.11b/g.

1.3.6 IEEE 802.11n

Tento standard, který vznikl v roce 2009, zavádí pro zvýšení přenosové rychlosti takzvanou metodu vícenásobného vstupu a výstupu (MIMO). MIMO funguje tak, že se datový tok rozdělí na několik menších, v tomto případě 4, které jsou pak vyslány každý přes samostatnou anténu. Na přijímací straně je potom stejný počet antén, aby bylo možno data přijmout. Díky tomu může zůstat zachována původní šířka pásma.

Tento standard operuje ve frekvenčních pásmech 2,4 i 5 GHz a podporuje jak DSSS tak OFDM, což mu umožňuje zpětnou kompatibilitu se standardy 802.11 a, b, g. Maximální rychlost se pohybuje okolo 600 Mb/s.

1.3.7 IEEE 802.11p

Tento standard byl definován pro použití v pohybujících se vozidlech ke komunikaci s infrastrukturou okolo. Použité frekvenční pásmo je 5,9 GHz a maximální přenosová rychost 27 Mb/s.

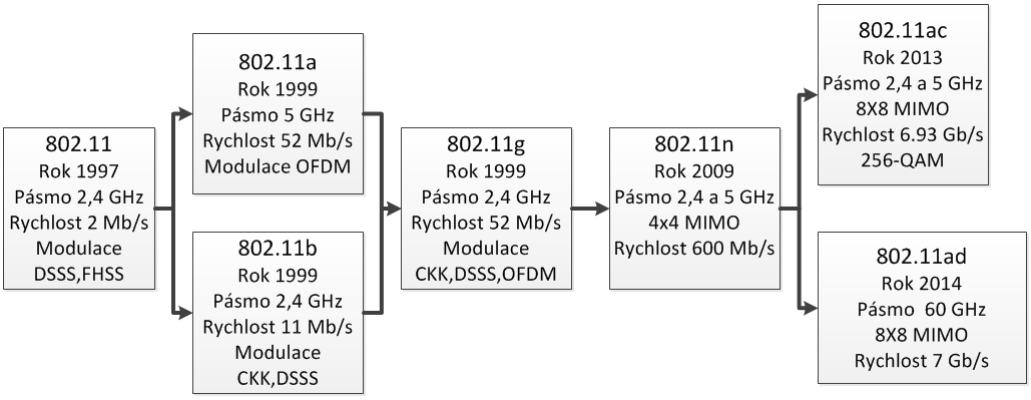
1.3.8 IEEE 802.11ac

Tento standard vyšel v roce 2013 a jedná se o rozšíření IEEE 802.11n. MIMO nyní podporuje rozdělení datového toku až na 8 dílčích. Signál je modulován hustší amplitudovou modulací 256-QAM (4x více než u verze n) a šířka pásma byla navýšena na 80 MHz (160 MHz). Teoretická maximální rychlost je 6,93 Gb/s.

1.3.9 IEEE 802.11ad

Tento standard byl uveden na trh v roce 2014. Místo frekvenčních pásem 2,4 a 5 GHz je zde využito pásmo na 60 GHz. Maximalní přenosová rychlost v pásmu 60 GHz je teoreticky 7 Gb/s, ale díky vysokému kmitočtu dochází ke snížení maximálního dosahu signálu na max 10 m (předchozí standardy mají přibližný dosah 30-50 m).

Výpočet vlnové délky pro 60 GHz *pásmo*
λ *vlnová délka*
v *rychlost* šíření *světla ve vakuu* (300000 Km/s)
f *frekvence*
$\lambda = \frac{v}{f} = \frac{300000000}{60000000000} = 0,005\ m$



Obr. 1.6: Přehled vývoje standardů v 802.11

2 WEP (WIRED EQUIVALENT PRIVACY)

Tento šifrovací algoritmus byl poprvé představen již v roce 1997 jako součást prvního standardu IEEE 802.11. Cílem bylo poskytnout přenosu dat stejnou bezpečnost, jako je v sítích ethernet. K ochraně dat využívá symetrickou proudovou šifru RC4. Vlastní šifrovací klíč se skládá z tajného klíče, který má délku 40 b nebo 104 b (5 nebo 13 znaků z abecedy ACSII), a inicializačního vektoru (dále IV) o délce 24 b. WEP jako takový nebyl navržen odborníky z oblasti kryptografie, což se následně projevilo množstvím slabých míst v rámci algoritmu. Informace k různým druhům zabezpečení Wi-Fi jsou na [4],[7] a [14].

2.1 Inicializační vektor (IV)

IV je blok bitů nutný pro kodér, aby bylo možné generovat unikátní toky šifrovacího textu při využití jednoho šifrovacího klíče. Z tohoto vyplývá, že IV musí znát i protistrana, jinak není možné v přijímači tuto zprávu dekódovat. Předání IV je možné realizovat několika způsoby:

- Dohodnutím se obou stran během výměny klíčů nebo při navázáni spojení („Handshake“).
- Přiložením IV k zašifrovanému textu.
- Výpočtem z různých parametrů, jako jsou časový údaj, adresy vysílače a přijímače, čísla paketu, atd.

V rámci WEP bylo použito metody přiložení IV ke zprávě. Navíc se použilo příliš krátkého IV (24 b), což vedlo k opakovanému použití stejných IV (Uvádí se 7 až 8 hodin k spořebování všech IV) a tedy k možnému riziku prolomení zabezpečení.

2.2 CRC32

Jedná se o kód na detekci náhodných chyb ve zprávě. Přidává k původní zprávě nadbytečnost o délce 32 b, která je vypočtena pomocí bitového dělení. Z hlediska šifrování je tento algoritmus nevhodný, protože díky tomu, že je lineární, je snadné pozměnit chráněná data.

2.3 RC4

RC4 je proudová šifra s variabilní délkou šifrovacího klíče. Díky své rychlosti a jednoduchosti se tato šifra stala volbou pro ochranu přenosů v sítích. RC4 je například

použita v SSL, WEP a u WPA jako součást TKIP. Tato šifra v sobě implementuje kombinaci dvou šifrovacích algoritmů, pseudonáhodného generátoru (PRGA) a algoritmu rozvrhnutí klíče (KSA).

2.3.1 KSA

```
for i od 0 do 255
   S[i] := i
end
j := 0
for i od 0 do 255
   j := (j + S[i] + key[i mod keylength]) mod 256
   prohození hodnot S[i] a S[j]
end
```

Obr. 2.1: Princip KSA

Nejdříve se vytvoří vektor S, který se postupně naplní čísly 0 až 255. Následně dojde k promíchání těchto čísel dle šifrovacího klíče. Získáme tedy vektor s čísly 0 až 255 pseudonáhodně proházenými.

2.3.2 PRGA

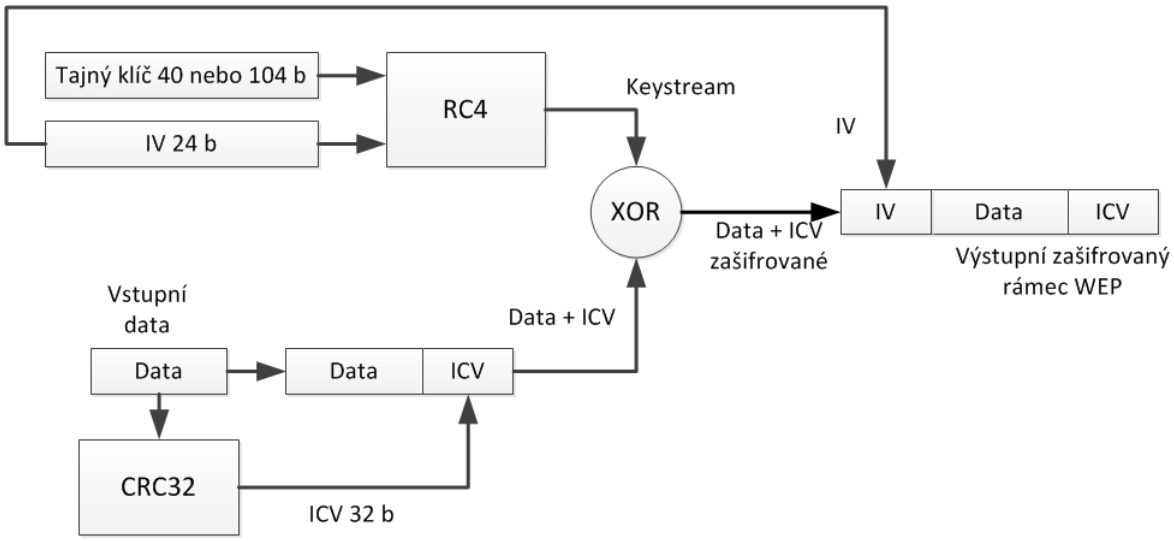
```
i := 0
j := 0
For z od 0 do délka rámce dat
   i := (i + 1) mod 256
   j := (j + S[i]) mod 256
   prohozeni hodnot S[i] a S[j]
   keystream[z] := S[(S[i] + S[j]) mod 256]

end
```

Obr. 2.2: Princip PRGA

Dle délky datového rámce v bajtech se provede tolik cyklů, aby výstupní keystream byl stejné délky. V každém cyklu se provede prohození čísel ve vektoru S a zapíše se do keystreamu jeden bajt z vektoru S. Toto se provádí, dokud délka keystreamu není stejná s délkou datového rámce.

2.4 Postup šifrování

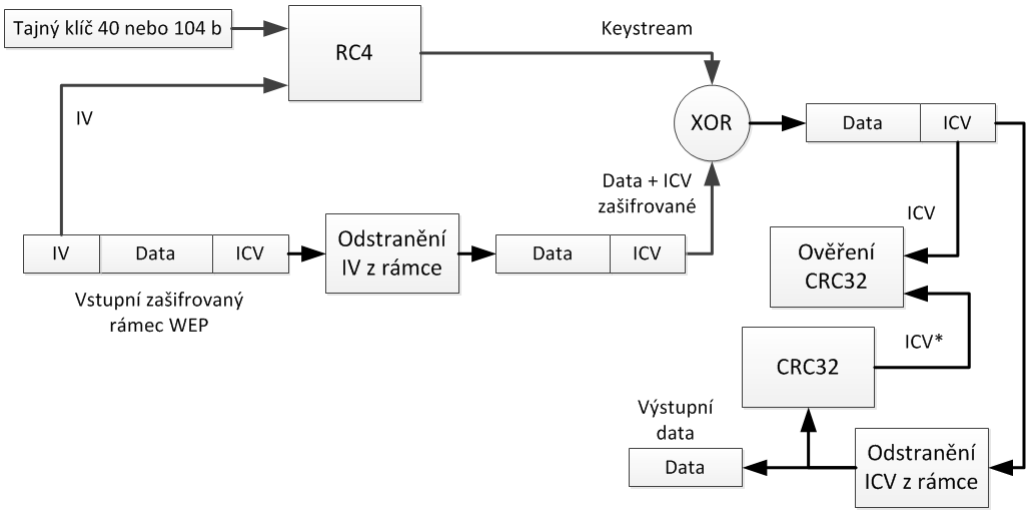


Obr. 2.3: Postup šifrování dat pomocí WEP

1. Vstupní rámec dat je pomocí CRC32 zabezpečen a výsledek CRC32 je přidán na konec datového rámce.
2. V bloku RC4 se sloučí tajný klíč s IV do 64 b (128 b) šifrovacího klíče a následně se tento klíč použije pro provedení permutací k vytvoření pseudonáhodné posloupnosti (keystream) na výstupu tohoto bloku. Tato posloupnost se ještě přizpůsobí, aby délka byla stejná s délkou vstupních dat s CRC32 a tedy bylo možné provést další operaci.
3. Keystream a vstupní data s CRC32 se sečtou za pomocí XOR operace.
4. Pro dokončení WEP rámce je ještě nutné připojit k rámci IV, čímž na výstupu dostáváme konečný tvar rámce.

2.5 Postup dešifrování

1. Ze vstupního rámce dat je oddělen IV.
2. IV a tajný klíč vytvoří v RC4 šifrovací klíč, který se použije pro vytvoření keystreamu.
3. Keystream a vstupní data bez IV se sečtou za pomocí XOR operace, na výstupu je dešifrovaný rámec dat s ICV.
4. Z rámce se oddělí ICV.



Obr. 2.4: Postup šifrování dat pomocí WEP

5. Nad datovým rámcem se provede CRC32 a výsledek ICV* se porovná s ICV. Pokud jsou stejné, datový rámec je vyhodnocen jako v pořádku, pokud nejsou, signalizuje se chyba vyšší vrstvě.

2.6 Slabá místa WEP

2.6.1 Autentizace

Tato chyba se vyskytuje ve WEP, pokud používáme autentizaci sdíleným klíčem. Důvodem je, že text odeslaný ke stanici od AP, který je vyžadován zašifrovat, se posílá nešifrovaně. Pokud tedy útočník zachytí tuto úvodní komunikaci i s odpovědí na ni ze stanice, získává tak všechny potřebné informace k získání keystreamu. Platí totiž, že šifrovaná data jsou výsledkem XOR operace mezi keystreamem a nešifrovanými daty a pro operaci XOR platí, pokud použijeme XOR mezi šifrovanými a nešifrovanými daty, získáme keystream. V tomto okamžiku zná útočník keystream a i bez znalosti vlastního tajného klíče může posílat do sítě vlastní pakety.

Označíme keystream jako K*, nešifrovaný text od AP jako* P *a zašifrovaný text odeslaný zpět stanicí jako* C*.*
Platí : $P \oplus K = C$
Z tohoto lze odvodit : $C \oplus P = (P \oplus K) \oplus P = (P \oplus P) \oplus K = K$

$\oplus$ *XOR operace*

2.6.2 Řízení přístupu

Kromě jednoduchého řízení přístupu pomoci sdíleneho klíče není tato funkce součástí WEP. Někteří výrobci přidali filtrování MAC adres, aby omezili počet povolených zařízení v síti. I toto řešení je relativně snadné obejít, protože hlavička paketu s MAC adresou se posílá nešifrovaně. Útočníkovi tedy stačí odposlechnout jediný paket, aby získal MAC adresu povoleného zařízení a následně změnil svoji MAC adresu na tuto získanou.

2.6.3 Neomezená platnost paketu

V systému WEP se toto neřeší a proto je platnost paketu neomezená. V praxi to znamená, že útočníkem odchycená i starší data mohou být použita jako ta aktuální a pak je možné je použít k útoku na systém.

V bezpečných systémech se používá počitadel nebo posuvného okénka, aby se zabránilo těmto útokům.

2.6.4 CRC32

Tento algoritmus není vhodný k použití pro šifrování, protože jeho primární použití je detekce nahodilých chyb vzniklých při přenosu dat. Poskytuje tedy jen jednoduchý algoritmus za tímto účelem, který je lehce spočitatelný, což útočníkovi umožňuje snadno upravit paket a následně změnit i výsledek CRC32. Díky tomu je pro útočníka snadné měnit obsah paketu.

2.6.5 Slabé tajné klíče

Pro některé klíče je snadnější odvodit jejich vazbu na zašifrovaný text. U 40 b tajného klíče se odhaduje okolo 9000 slabých kombinací znaků. Navíc 40 b klíč je relativně krátký a tedy pro moderní počítače není obtížné ho prolomit pomocí „brute force“ útoku, tento problém je možné vyřešit použitím 104 b klíče. V současnosti ale existují i sofistikovanější metody, které mění exponenciální závislost času potřebného na prolomení klíče na lineární.

$40\ b\ klíč : 2^{40} = 1,1 * 10^{12}\ kombinací$
$104\ b\ klíč : 2^{104} = 2 * 10^{30}\ kombinací$

2.6.6 IV

Díky nevhodné implementaci IV, kdy IV je posíláno nešifrovaně jako součást zašifrovaného paketu, může útočník okamžitě po odchycení paketu získat IV a tedy mít k dispozici 24 b z šifrovacího klíče. Další nevýhodou IV je jeho délka, protože 24 b je pro použití šifry RC4 příliš krátké. V praxi to znamená, že po relativně krátké době (7 až 8 hodin) se začnou vyskytovat pakety se stejným IV. Pokud útočník získal dva pakety se stejným IV, je možné pomocí operace XOR získat šifrovací sekvenci a to může vést k prozrazení obsahu zprávy.

2.6.7 RC4

V RC4 dochází k problému s vygenerovaným keystreamem. Tento nedostatek souvisí s odesíláním nešifrovaného IV u každé zprávy a výsledkem je, že útočník může předvídat následující bajt keystreamu.

Kromě této slabiny se vyskytuje jestě slabé místo v generování keystreamu ze slabých klíčů, kde platí, že slabý klíč generuje slabý keystream. Pro tento problém existuje řešení v podobě odstranění určitého počtu úvodních bajtů z posloupnosti.

3 WPA

U šifrování pomocí WEP se ukázalo, že obsahuje mnoho slabých míst, která je možné zneužít. Proto bylo v roce 2003 představeno nové řešení WPA (Wireless Protected Access). Původní řešení WPA bylo navrhnuto tak, aby bylo možné ho použít na všech zařízeních, která podporují WEP, pouze s úpravami v softwarové části. Bylo čerpáno těchto zdrojů [1], [4], [6], [7] a [14].

3.1 Rozdíly proti WEP

WEP kodér je rozšířen na TKIP, kdy WEP část zůstává stejná, ale přidaly se algoritmy řešící problémy WEP. Došlo k nahrazení nevyhovujícího CRC32 novým algoritmem s názvem MIC (Message Integrity Check), který je generován pomocí kryptografického algoritmu Michael. Přidání počítadla TSC (TKIP Sequence Counter) řeší problém neomezené platnosti v WEP. TSC je prodloužené IV z WEP. Zavedením odvozování klíče pro každý paket se odstranil problém s přímým použitím šifrovacího klíče a použitím slabých klíčů. IV bylo prodlouženo na 48 bitů.

3.1.1 TSC (IV)

Došlo k prodloužení délky TSC na dvojnásobek z 24 b WEP na 48 b. Nyní se již TSC nepřipojuje přímo k šifrovacímu klíči, ale používají se algoritmy, kde TSC slouží k odvozování šifrovacího klíče. TSC vždy začíná od hodnoty 0 a každý rámec tuto hodnotu inkrementuje o 1. TSC je vloženo do WPA rámce v nešifrované podobě. Na přijímací straně je zavedena kontrola přijatého TSC proti hodnotě v paměti přijímače. Pokud TSC hodnota přijatého rámce je nižší než hodnota v paměti přijímače, rámec je zahozen.

3.1.2 TKIP (Temporal Key Integrity Protocol)

Tento algoritmus byl zaveden z důvodu ochrany tajného klíče. Místo přímého použití tajného klíče se provede odvození paketového klíče z tajného klíče, TSC a MAC adresy vysílače. Paketový klíč se počítá pro každý rámec samostatně a tedy každý paketový klíč je jiný než předchozí.

3.1.3 MIC (Message Integrity Check)

Toto řešení bylo implementováno jako náhrada za CRC32 pro zlepšení ochrany před falšování rámců. MIC se počítá z algoritmu Michael, který poskytuje lepší zabezpečení než původní CRC32. Výpočet se skládá z pravé a levé rotace, sčítání *modulo* 2^{32}

a XOR operací. Data, nad kterými se počítá MIC, jsou cílová a zdrojová adresa a vlastní data.

3.2 Čtyřcestné podání rukou (4-way handshake)

Tato metoda je modifikovanou metodou autentizace sdíleným klíčem, která odstraňuje možnost snadného odhalení klíče, jak tomu bylo u WEP. Toto řešení autentizace stanice a přístupového bodu se používá u WPA a WPA2. Autentizace touto metodou funguje takto:

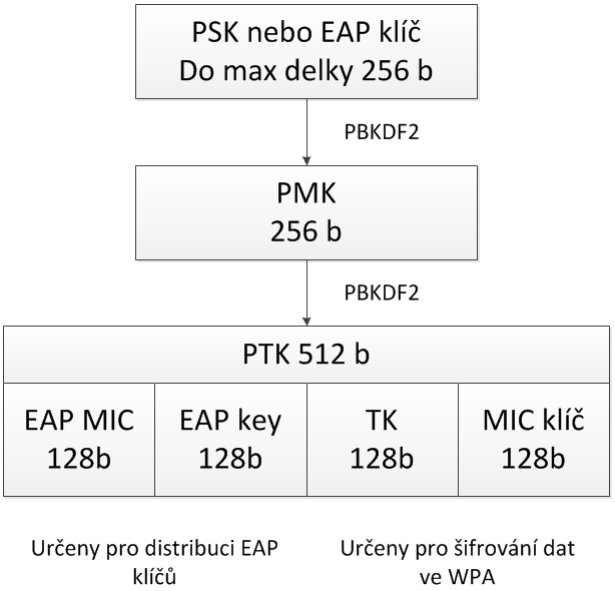
1. Stanice posílá žádost o připojení do sítě. Po přijetí této zprávy v AP se vytvoří ANonce, což je náhodné číslo, kterým AP započne vlastní autentizaci.
2. Po přijetí ANonce si stanice vytvoří také náhodné číslo SNonce. Tyto čísla se pak použijí k výpočtu párového přenosového klíče z tajného primárního klíče (PMK). SNonce je pak odesláno k AP s podpisem od PMK, ale samo jím není šifrováno.
3. AP přijme SNonce od stanice a protože zná primární klíč a má původní ANonce, může vypočítat přenosový klíč. Následně AP ověří integritu podpisu na SNonce paketu a tím zjistí, jestli stanice zná správný klíč. Pokud je vše v pořádku, odešle povolení ke stanici, které je podepsané přenosovým klíčem.
4. Stanice přijme toto povolení a ověří integritu podpisu. Když je vše v pořádku, tak stanice ví, že i AP má správný klíč.

Výhodou využití této metody je, že nepoužíváme přímo tajný klíč a tím se snižuje nebezpečí jeho vyzrazení. Podpis při výměně nonce zpráv je generován pomocí PBKDF2 hashovacího algoritmu, kdy nonce se pomocí něho zašifruje a PMK slouží jako heslo.

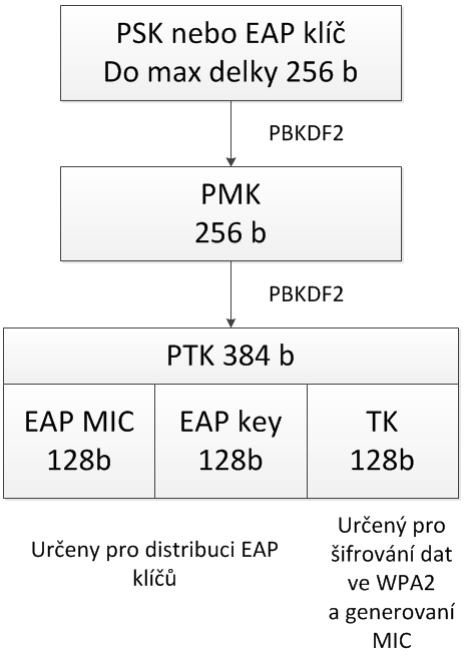
3.3 Rozlišení klíčů ve WPA a WPA2

1. Předsdílený klíč (PSK) - Tajný klíč umístěný ve stanici a AP.
2. Párový primární klíč (PMK) - Klíč derivovaný z PSK nebo z klíče zaslaného pomocí EAP protokolu.
3. Párový přenosový klíč (PTK) - Odvozený klíč z hlavního klíče, ANonce a SNonce, který se vytvoří ve stanici a AP při úspěšném dokončení autentizace. Tento klíč slouží k šifrováni jednotlivých paketů.
4. Dočasný klíč (TK) - 128 b klíč, jež je částí PTK.
5. Skupinový dočasný klíč (GTK) - Tento klíč je používán k adresaci všech prvků v rámci sítě.

6. MIC klíč - Klíč určený k použití v MIC, jedná se o dva klíče o délce 64b, kdy první je určen pro komunikaci z AP na stanici a druhý v obráceném směru. U WPA2 tento klíč neexistuje, protože je stejný s TK.



Obr. 3.1: Hierarchie klíčů ve WPA



Obr. 3.2: Hierarchie klíčů ve WPA2

3.4 PBKDF2

Jedná se pseudonáhodnou funkci používanou ve WPA a WPA2 zabezpečení, kde slouží k autentizaci prvků bezdrátové sítě a k hashování šifrovacích klíčů. Vnitřní funkce použitá k hashování klíčů je HMAC-SHA1. Oproti předchozí funkci PBKDF1 došlo k vylepšení, které umožňuje si zvolit délku výstupního klíče. Informace byly čerpány z [11]. Výpočet PBKDF2 je následující:

```
DK = PBKDF2 (text, salt, c, dkLen)
DK - výstupní hashovaný text
text - vstupní data, která mají být zašifrována
salt - klíč použitý k šifrování textu
c - číslo definující počet iteračních cyklů
dkLen - délka výstupního hashovaného textu definovaná v bajtech

n = CEIL(dkLen/hLen) definuje počet cyklů pro zadanou délku výstupu
 zaokrouhluje se na nejbližší vyšši integer

T1 = F (text, salt, c, 1)   T1 až Tn jsou segmenty DK
T2 = F (text, salt, c, 2)
...
Tn = F (text, salt, c, n)

kdy F(text, salt, c, n) = U1 xor U2 ... xor Uc

U1 až Uc jsou počítány dle HMAC-SHA1, i je hodnota
cylku tedy od 1 do n, tato hodnota se připojuje
k salt.

U1 = HMAC-SHA1(text, salt || INT_32(i))
U2 = HMAC-SHA1(text, U1)
...
Uc = HMAC-SHA1(text, U(c-1))

výstup je vytvořen spojením všech T1 až Tn
DK = (T1 || T2 ... || Tn)
```

3.5 HMAC

Tento algoritmus je určen ke generování autentizačního kódu zprávy za použití hashovací funkce. V případě WPA a WPA2 se jedná o SHA1. HMAC je definováno v [12]. Postup pro generování HMAC je následující:

1. Provede se doplnění klíče K o nuly do délky bloku, v případě WPA, WPA2 je délka tohoto bloku 64 B.
2. Opad, vnější padding, s hodnotou 0x5c a Ipad, vnitřní padding, s hodnotou 0x36 hexadecimálně se zopakují tolikrát, aby jejich nová délka odpovídala délce bloku.
3. Vytvoříme nové vektory Opad a Ipad provedením XOR operace původních Ipad a Opad s klíčem K.
4. Připojíme vstupní blok textu k Ipad a zavoláme SHA1 funkci.
5. Výsledek SHA1 se připojí k Opad a provedeme opět SHA1 funkci.
6. Hodnota vrácená SHA1 se poté pošle na výstup.

Postup uvedený výše lze složit do tohoto zápisu:

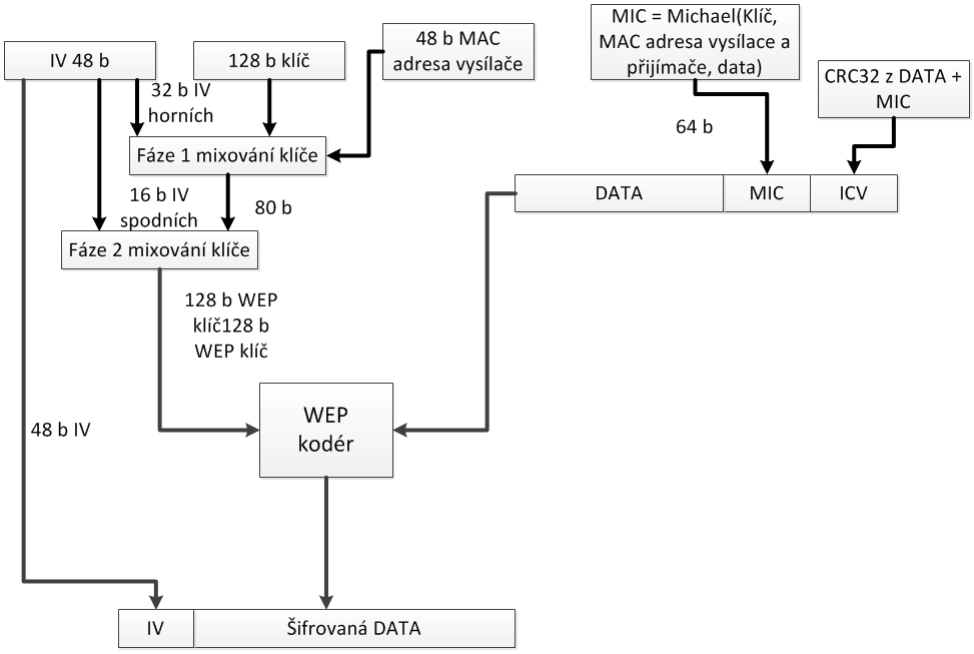
```
výstup = SHA1(K XOR Opad || SHA1(K XOR Ipad || text))
```

3.6 SHA1

SHA1 patří mezi světově nejrozšířenější hashovací algoritmy. Tento algoritmus je schopen kromě hashovaní zpráv zajistit i jejich datovou integritu. SHA1 je popsána ve standardu [5], kde je dispozici i pseudokód realizace. Každá hashovací funkce, která má jít do veřejného použití, musí být zkontrolována proti testovacím vektorům, které definoval národní institut standardů a technologie v USA. Pro Českou republiku tyto normy přebírá a upravuje Úřad pro technickou normalizaci, metrologii a státní zkušebnictví. V současnosti existuje více než 2000 ověřených implementací.

3.7 TKIP

Klíč je vypočítán z tajného klíče a SSID pomocí pseudonáhodné funkce PBKDF2. Nejprve je nutné z cílové a zdrojové MAC adresy, klíče a dat vypočítat MIC, které připojí k datovému rámci. Rámec dat s MIC je zabezpečen proti chybám pomocí CRC32. Z horních 32 b IV, klíče a MAC adresy vysílače se pomocí fáze 1 mixování klíče vytvoří nový 80 b klíč. Tento klíč se sloučí se spodními 16 b IV a provede se fáze 2 mixování klíče. Výsledkem je 128 b WEP klíč, tímto klíčem se pak zašifruje



Obr. 3.3: Postup šifrování TKIP

pomocí původního WEP kodéru rámec dat s MIC a ICV. Postup šifrování v TKIP je možné vidět na obr. 3.3.

3.7.1 Substituční tabulka S-BOX

Substituční tabulka slouží k provádění substitucí pro obě fáze mixování klíče. Pro snadnější použití a zrychlení operací bývá umístěna v paměti zařízení. Protože celá tabulka je definována jako matice 256x256 16 bitových prvků, v praxi se používají 2 tabulky o celkové délce 512 8 b prvků. Vyhledávání v této tabulce probíhá nejprve rozdělením hledaného slova na horní a spodní bajt, kdy první bajt slouží k indexování v první tabulce a druhý v druhé. Následným spojením bajtů získaných ze substitučních tabulek dostaneme 16 b hodnotu pro použití v dalších operacích. Substituční tabulky dle standardu jsou v příloze B.1.

3.7.2 Fáze 1 mixování klíče

V této fázi se použije 128 b klíč, MAC adresa vysílače a horních 32 b TSC, ze kterých se vytvoří 80 b klíč pro další fázi mixování klíče. Tento klíč se skládá z bloků P1K0, P1K1, P1K2, P1K3, P1K4, kdy každý z nich je dlouhý 16 b. V této fázi se používá pouze sčítání, XOR operací a spojování dvou 1 B slov do 2 B slov. Pseudokód postupu mixování je na obr. 3.4.

```
PHASE1_STEP1
 P1K0 = TSC1
 P1K1 = TSC2
 P1K2 = TA1∩TA0
 P1K3 = TA3∩TA2
 P1K4 = TA5∩TA4

PHASE1_STEP2:
 FOR i = 0 to 3
 BEGIN
   P1K0 = P1K0 + S[ P1K4 ⊕ (TK1∩TK0) ]
   P1K1 = P1K1 + S[ P1K0 ⊕(TK5∩TK4) ]
   P1K2 = P1K2 + S[ P1K1 ⊕(TK9∩TK8) ]
   P1K3 = P1K3 + S[ P1K2 ⊕(TK13∩TK12) ]
   P1K4 = P1K4 + S[ P1K3 ⊕(TK1∩TK0) ] + i
   P1K0 = P1K0 + S[ P1K4 ⊕(TK3∩TK2) ]
   P1K1 = P1K1 + S[ P1K0 ⊕(TK7∩TK6) ]
   P1K2 = P1K2 + S[ P1K1 ⊕(TK11∩TK10) ]
   P1K3 = P1K3 + S[ P1K2 ⊕(TK15∩TK14) ]
   P1K4 = P1K4 + S[ P1K3 ⊕(TK3∩TK2) ] + 2*i + 1
END
```

TK0-TK15 Bajty vstupního klíče
0 - nejnižší 15 - nejvyšší
TSC1 Prostředních 16 b TSC
TSC2 Horních 16 b TSC
TA0-TA5 Bajty MAC adresy vysílače
∩ kombinace bajtů do 16 b slova
x∩y = 256*x + y
S[] Substituce
⊕ Operace XOR

Obr. 3.4: Pseudokód fáze 1 mixování klíče

3.7.3 Fáze 2 mixování klíče

Z 80 b klíče z předchozí fáze a spodních 16 b TSC se vytvoří paketový klíč o délce 128 b pro RC4 blok. Postup je uveden na obr. 3.5.

3.7.4 MIC

Vstupní data se se musí nejprve upravit do vhodného tvaru, než je možné použít vlastní MIC operace. V prvním kroku se k datům na konec připojí bajt s hodnotou 0x5A. Následně je k datům připojeno tolik nulových bajtů, aby data byla dělitelná 4 v bajtech. Minimální počet připojených nulových bajtů jsou 4. Tyto přidané bajty jsou přidány pouze pro výpočet MIC a přes linku se nevysílají. Takto upravená data jsou připravena ke zpracování MIC algoritmem. Výsledkem MIC výpočtu jsou dvě 32 b hodnoty V0 a V1, které dohromady dávají 64 b MIC pro rámec před šifrováním. Pseudokód pro MIC blok je na obr. 3.6.

3.8 Dešifrování TKIP

Od přijatého rámce se oddělí TSC (IV) a ověří se, jestli TSC přijatého rámce má vyšší hodnotu než TSC v paměti příjemce. Následně se vytvoří pomocí TKIP WEP klíč pro WEP dekodér, kdy postup je stejný jako při šifrování. WEP dekodér nyní

```
PHASE2,STEP1:
    PPK0 = P1K0
    PPK1 = P1K1
    PPK2 = P1K2
    PPK3 = P1K3
    PPK4 = P1K4
    PPK5 = P1K4 + TSC0

PHASE2,STEP2:
    PPK0 = PPK0 +  S[ PPK5 ⊕(TK1∩ TK0) ]
    PPK1 = PPK1 +  S[ PPK0 ⊕(TK3∩ TK2) ]
    PPK2 = PPK2 +  S[ PPK1 ⊕(TK5∩ TK4) ]
    PPK3 = PPK3 +  S[ PPK2 ⊕(TK7∩TK6) ]
    PPK4 = PPK4 +  S[ PPK3 ⊕(TK9∩TK8) ]
    PPK5 = PPK5 +  S[ PPK4 ⊕(TK11∩TK10) ]
    PPK0 = PPK0 +  >>>(PPK5 ⊕(TK13∩TK12))
    PPK1 = PPK1 +  >>>(PPK0 ⊕(TK15∩TK14))
    PPK2 = PPK2 +  >>>(PPK1)
    PPK3 = PPK3 +  >>>(PPK2)
    PPK4 = PPK4 +  >>>(PPK3)
    PPK5 = PPK5 +  >>>(PPK4)
PHASE2,STEP3:
    RC4Key0 =  UpperByte(TSC0)
    RC4Key1 =  (UpperByte (TSC0) | 0x20) & 0x7F
    RC4Key2 =  LowerByte(TSC0)
    RC4Key3 =  LowerByte ((PPK5 ⊕>(TK1∩TK0))
    RC4Key4 =  LowerByte (PPK0)
    RC4Key5 =  UpperByte (PPK0)
    RC4Key6 =  LowerByte (PPK1)
    RC4Key7 =  UpperByte (PPK1)
    RC4Key8 =  LowerByte (PPK2)
    RC4Key9 =  UpperByte (PPK2)
    RC4Key10 = LowerByte (PPK3)
    RC4Key11 = UpperByte (PPK3)
    RC4Key12 = LowerByte (PPK4)
    RC4Key13 = UpperByte (PPK4)
    RC4Key14 = LowerByte (PPK5)
    RC4Key15 = UpperByte (PPK5)
```

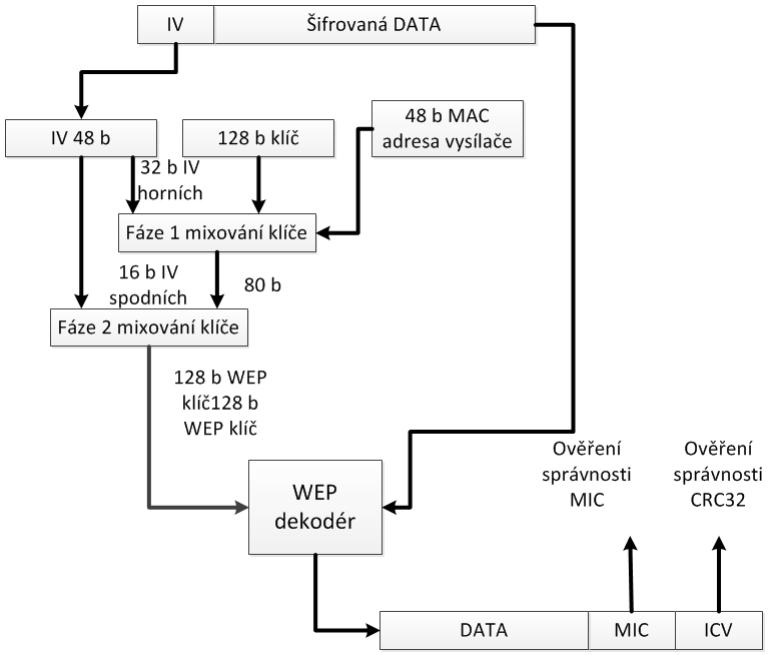
TK0-TK15 Bajty vstupního klíče
0 - nejnižší 15 - nejvyšší
TSC0 Dolních 16 b TSC
∩ kombinace bajtů do 16 b slova
x∩y = 256*x + y
S[] Substituce
⊕ Operace XOR
| Operace OR
& Operace AND
>>> rotace 16 b slova o 1 vpravo
> posunutí 16 b slova o 1 vpravo
RC4key0 – 15 výstupní bajty paketového klíče

Obr. 3.5: Pseudokód fáze 2 mixování klíče

```
l = K0
r = K1
for i=0 to N-1
l = l ⊕ Mi
r =  r ⊕ (l <<< 17)
l = (l + r) mod 2^32
r = r ⊕ XSWAP(l)
l = (l + r) mod 2^32
r = r ⊕ (l <<< 3)
l = (l + r) mod 2^32
r = r ⊕ (l >>> 2)
l = (l + r) mod 2^32
end
V0 = l
V1 = r
```

```
K0, K1 MIC klíč rozdeleny na 2x 32 b
N Počet bloků textu
Mi i-ty 32b blok textu
⊕ Operace XOR
 >>> rotace 32 b slova  vpravo
<<<  rotace 32 b slova  vlevo
V0, V1 Výstupní MIC
XSWAP Prohozeni dolnich a hornich
16 b
```

Obr. 3.6: Pseudokód generování MIC



Obr. 3.7: Postup dešifrování TKIP

již s klíčem dešifruje přijatá data. Nakonec se ověří správnost CRC32 a MIC v dešifrovaném rámci. Postup je ukázán na obr. 3.7.

3.9 Slabá místa WPA

3.9.1 Slabé klíče

Stejně jako u WEP platí, že zvolením slabého tajného klíče zjednodušujeme útočníkovi práci s prolomením hesla. Délka hesla je 8 až 63 ASCII znaků a za bezpečné se považuje heslo o délce nad 20 znaků.

3.9.2 WPS

WPS (Wi-Fi protected setup) bylo vytvořeno pro zjednodušení konfigurace nových prvků ve Wi-Fi sítích, ale ukázalo se jako velké bezpečnostní riziko. WPS je sestaveno z 8 dekadických číslic, které bylo možné pomocí „bruteforce“ útoků do několika hodin dešifrovat. Toto útočníkovi umožní nejenom připojit se do sítě, ale také získat WPA či WPA2 klíč.

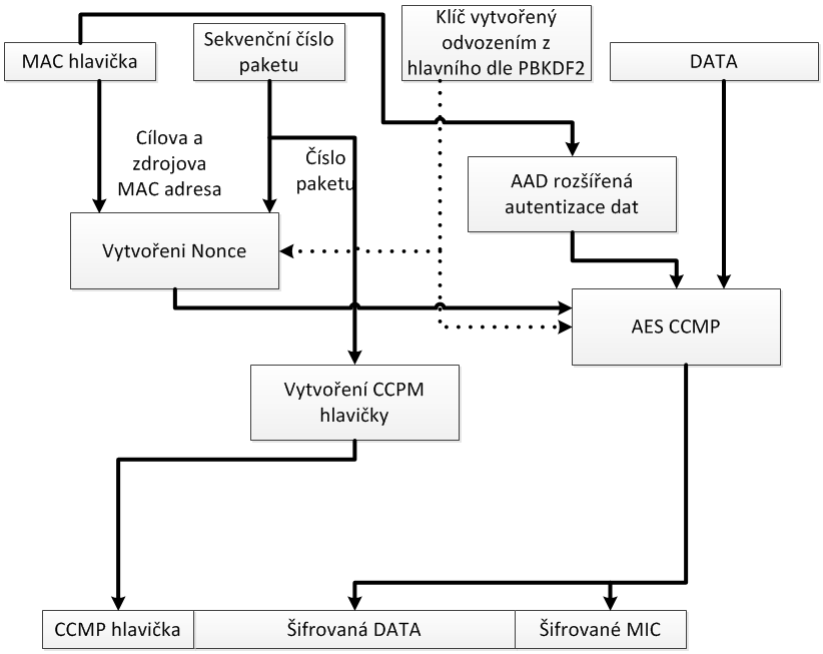
3.9.3 QoS

Díky použití QoS došlo k možnosti, že stejný keystream je možné opakovaně použít v různých frontách, protože každá fronta má vlastní sekvenční počítadlo. Tohoto je možné použít u krátkých paketů, jako jsou ARP, kde je možné z tohoto paketu dešifrovat MIC a následně provést injektáž vlastního paketu. Tento útok se nazývá „Beck and Tews“. Jelikož tato injektáž funguje pouze s krátkými pakety, jako jsou ARP, závisí na použití QoS a šifrovací klíč stále zůstává tajný, nejedná se o zásadní bezpečnostní riziko.

4 WPA2

Jedná se o vylepšení WPA, které nahrazuje původní TKIP využívající RC4 novým řešením, jež se nazývá pokročilý šifrovací standard (AES). WPA2 bylo uveřejněno v roce 2004 a do současnosti nebyla nalezena žádná zásadní chyba, která by vedla k zásadnímu bezpečnostnímu riziku. Oproti WPA je zde snaha redukovat data přidaná šifrovacími algoritmy, proto má oproti WPA vyšší propustnost užitečných dat. Díky velké složitosti WPA2 vyžaduje pro provedení vyšší výpočetní výkon než předchůdci. Bylo čerpáno těchto zdrojů [1], [8], [9], [10]

4.1 Šifrování v WPA2

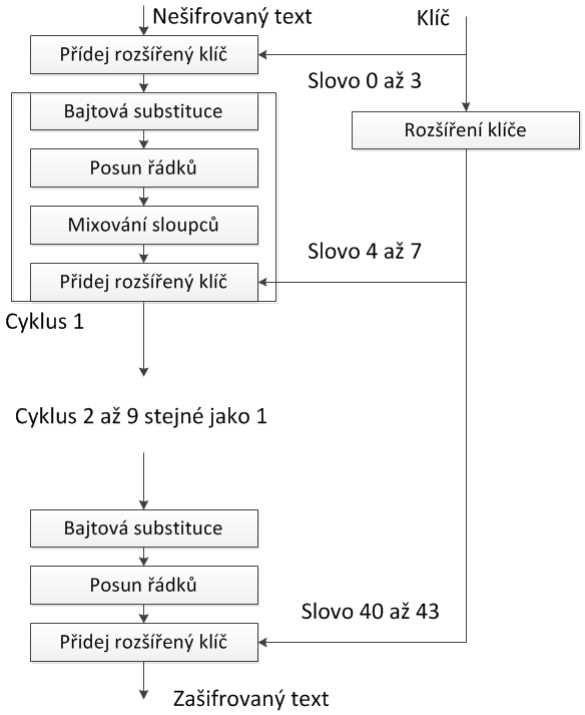


Obr. 4.1: Diagram šifrování ve WPA2

V prvním kroku se zvýší číslo paketu a přes MAC hlavičku se provede AAD rozšířená autentizace dat, která slouží k zabezpečení MAC hlavičky. Vytvoří se Nonce pro paket a vše se odešle do bloku AES CCMP. AES CCMP slouží k vlastnímu zašifrování dat a přidání MIC vypočítaného pomocí AES. Výsledný rámec sestává z CCMP hlavičky a šifrovaných dat s MIC.

4.2 AES

AES je symetrickou blokovou šifrou, která je jádrem WPA2. Pro účely WPA2 je délka bloku definovaná na 128 b a počet cyklů na 10. Bloky jsou organizovány do matice 4x4, kdy každý prvek matice je tvořen 8 bity.



Obr. 4.2: Diagram šifrování AES

Algoritmus začíná připojením prvního rozšířeného klíče, následně se provede 9 cyklů bajtové substituce, posunu řádků, mixování sloupců a přidání rozšířeného klíče. Cyklus 10 se provádí stejně jako předchozí cykly jen s vynecháním operace mixováni sloupců. Po provedení těchto operací je na výstupu šifrovaný text.

Dešifrování probíhá stejným způsobem jako šifrování, pouze v obráceném pořadí s inverzními funkcemi pro substituci, posun řádků a mixování sloupců.

Bajtová substituce

V AES je vytvořena substituční matice 16x16 bajtů, ze které jsou vyčteny jednotlivé prvky dle hodnoty vstupního parametru funkce. Prvni 4 bity se použijí jako adresa řádku a druhé 4 bity jako adresa sloupce.

Dle tohoto principu je postavena i inverzní bajtová substituce.

Posun řádků

Tato operace posunuje postupně prvky v jednotlivých řádcích. První řádek zůstává stejný, druhý je rotován o bajt doleva, třetí je rotován o 2 bajty doleva a u posledního je to o 3 bajty doleva.

Inverzní operací je posun řádků o stejné počty bajtů doprava.

Míchání sloupců

V této operaci se provádí substituce každé hodnoty každého sloupce matice jako funkce všech ostatních prvků v rámci sloupce. Výpočty jsou prováděny v Galloisově poli 2^8. Pro matici 4x4 se počítá operace míchání sloupců následujícím způsobem:

$$s^1_{0,j} = (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s^1_{1,j} = s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j}$$
$$s^1_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})$$
$$s^1_{3,j} = (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})$$

$\bullet$ *je násobení v Galloisově konečném poli* $GF(2^8)$
$s^1_{i,j}$ *jsou prvky nové matice,* $s_{i,j}$ *jsou prvky původní matice*

Násobení v Galloisově poli znamená, že bajty, které násobíme, převedeme na jejich matematický polynomiální tvar a vynásobíme je mezi sebou. Tímto získáme nový polynom, pokud se v tomto polynomu nevyskytuje mocnina vyšší než 7 je toto výsledek. Pokud je zde mocnina vyšší než 7, je nutné navíc tento polynom vydělit neredukovatelným polynomem $m(x) = x^8 + x^4 + x^3 + x + 1$. Tento polynom byl definován tvůrci AES šifry.

Přidání rozšířeného klíče

Jedná se o bitový XOR mezi 128 b stavem z předchozího bloku a 128 b rozšířeného klíče na příslušném řádku.

Rozšíření klíče

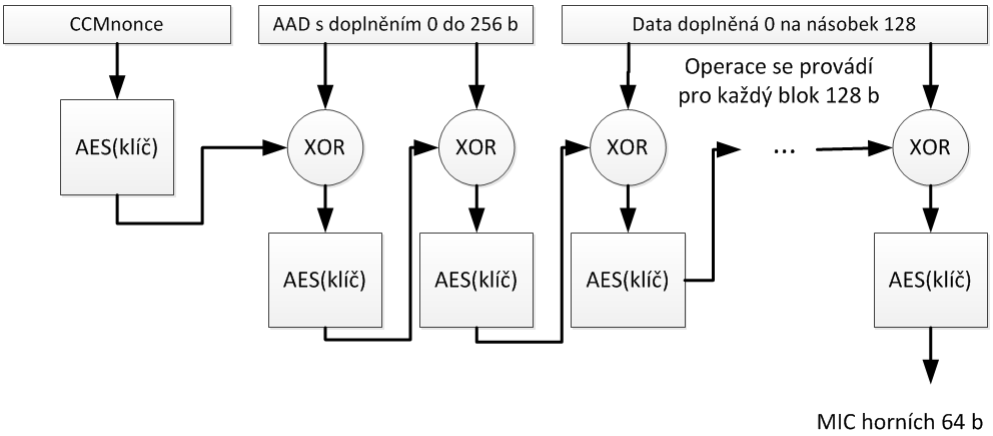
Rozšíření klíče vytváří z původních 4 slov o délce 128 b lineární pole o 44 slovech. Tedy každý z rozšířených klíčů má délku 128 b. První 4 slova jsou stejná jako původní. Následující slova jsou výsledkem provedení rotace o 8 b doleva, každý bajt slova je nahrazen dle substituce pomocí S-Boxu novým bajtem. Výsledek je pomocí XOR s Rcon, což je řádková konstanta, kde pouze první bajt nese hodnotu, převeden na finální klíč pro daný blok AES šifry.

S-Box

Podobně jako u WPA je i ve WPA2 zavedena tabulka pro substituci. Tato tabulka je kratší než u předchůdce, obsahuje pouze 256 záznamů. Celá tabulka je generována pomocí inverzního násobení v Galloisově poli dle neredukovatelného polynomu. Pro dešifrování je nutné použít inverzní S-Box, který je inverzní k puvodní tabulce.

4.3 MIC

MIC je nyní počítáno s pomocí AES šifry. Postup je následující: výsledek předchozího bloku šifrovaného AES se pomocí operace XOR spojí s dalším blokem dat a opět se provede AES šifrování. Součástí dat, nad kterými se provádí výpočet MIC, je i CCMnonce a AAD. Toto se opakuje, dokud nejsou spotřebovány všechny vstupní bloky. MIC je poté bráno jako 64 nejdůležitějších bitů z výstupu posledního bloku AES. Tento postup je možné vidět na obr. 4.3.



Obr. 4.3: Diagram vytvoření MIC

4.3.1 CCMnonce

Návěstí 1 B	Priorita (QoS) 1 B	MAC adresa zdroje SA 6 B	PN (IV) 6 B	dLen 2 B

Obr. 4.4: Struktura CCMnonce

Struktura CCMnonce je na obr. 4.4. Pole priorita definuje, jaká třída zacházení s rámcem je nastavena pro data dle QoS, pokud není použito QoS, je toto pole naplněno nulami. PN definuje číslo rámce. Tato hodnota je inicializována na 0 při začátku spojení a každý rámec ji inkrementuje o 1. PN odpovídá funkci TSC u WPA. Pole dLen definuje délku dat v bajtech v přenášeném rámci. Struktura návěstí je následující:

- bit 7 – rezervováno
- bit 6 – nastaven na 1, značí, že MIC se počítá i přes hlavičku rámce.
- bit 3-5 – definuje délku MIC, nastaveno na dekadickou hodnotu 3 pro délku 8 B MIC
- bit 0-2 – určuje délku dLen v bajtech, nastaveno na dekadickou hodnotu 1 pro dLen 2 B.

4.3.2 AAD

Tato část slouží k ochraně hlavičky rámce před úpravami, proto struktura tohoto bloku odpovídá původní MAC hlavičce pouze s vypuštěním částí, které se mohou měnit. Takovou částí je například doba trvání rámce. Tento blok je nutné prodloužit do délky 256 b, aby ho bylo možné zpracovat pomocí AES šifrování. Toto se provede přidáním nulových bitů pro doplnění do délky 256 b.

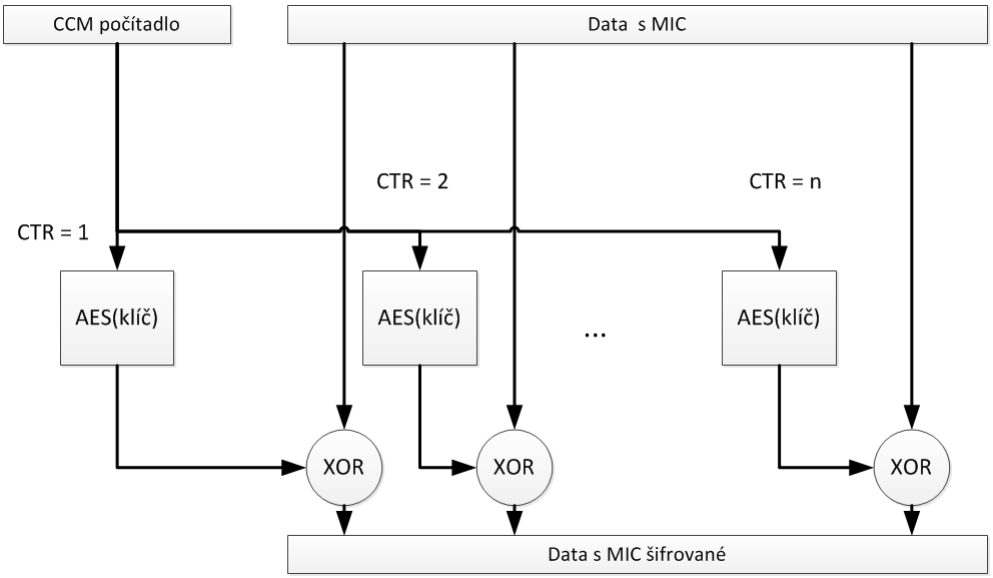
4.4 CCM počítadlo

Struktura CCM počitadla se podobá CCMnonce. Návěstí nyní obsahuje nuly kromě bitu 0, který je nastaven na hodnotu 1. Poslední 2 B puvodně vyhrazené pro dLen jsou zde použity jako počítadlo CTR. Hodnota počítadla se navyšuje o 1 s každým blokem dat. Průběh šifrování za pomoci počítadla je zobrzeno na obr. 4.5. Na každé CTR připadá 1 segment dat o délce 128 b, pokud poslední blok dat je kratší než 128 b je počítadlo zkráceno po průchodu AES šifrováním na stejnou délku. Tímto se odstraňuje problém blokových šifer, kdy je nutnost doplnit data na délku bloku.

4.5 Slabá místa WPA2

4.5.1 Slabé klíče

Podobně jako u předchozích zabezpečení i zde platí, že nevhodně zvolené heslo zjednodušuje útočníkovi možnost prolomit heslo. Vhodné heslo je 20 náhodných znaků dlouhé.



Obr. 4.5: Diagram šifrování dat za použití počitadla

4.5.2 SSID

Tato hodnota je přednastavena výrobcem routeru a je vhodné ji změnit, protože pro výrobcem definované hodnoty existují již vypočítané tabulky, které zjednodušují práci utočníkovi. Seznam aktuálně nejpoužívanějších SSID je možné nalézt na těchto stránkách [3].

4.5.3 WPS

Jedná se o stejný problém jako u WPA.

4.5.4 Hole196

Jedná se o útok, kdy útočník, pokud zná skupinový klíč GTK, má možnost odcizit identitu jiného uživatele uvnitř sítě. Jedná se o takzvaný *man-in-the-middle* útok, kdy útočník přesměrovává přes vlastní stanici komunikaci mezi napadeným uživatelem a přístupovým bodem a tedy odposlouchává celou jejich komunikaci. Další možností, jak tuto chybu lze využít, je bránění použití některé ze služeb útokem nazývaným *denial-of-service*. I tento útok je možné provést pouze na stanice nacházející se v rámci sítě, v které známe GTK.

5 TVORBA MODELŮ V PROSTŘEDÍ MATLAB SIMULINK

5.1 Matlab Simulink

Použitá verze Matlabu byla R2012b. Návod k Matlabu je k dispozici na [15]. Bylo zvoleno používání bloků v Matlabu s možností pracovat s celými rámci, protože ve skutečných systémech se nejprve skládají celé rámce, které se až poté odesílají bit po bitu po přenosovém médiu. Navíc řešení zobrazení jako rámců dat zvyšuje přehlednost celého modelu a zjednodušuje možnost prohlížení dat.

Proměnné v Simulinku byly definovány jako binární sloupcové vektory a proměnné pro blok `data store memory` jako dekadické. Jejich výběr byl zvolen z důvodu neschopnosti prostředí Simulink posílat písmenné proměnné skrze signálové cesty, což znemožňuje použití hexadecimálních a textových proměnných v těchto signálových cestách. Pokud by tedy bylo třeba, je možné text převést na binární formu pomocí příkazu `dec2bin('abc')` v Matlabu (každý znak v uvozovkách je převeden na řádek binárních hodnot dle ASCII tabulky, počet řádků odpovídá počtu znaků). Následně lze tyto hodnoty zkopírovat do vybrané proměnné v Simulinku.

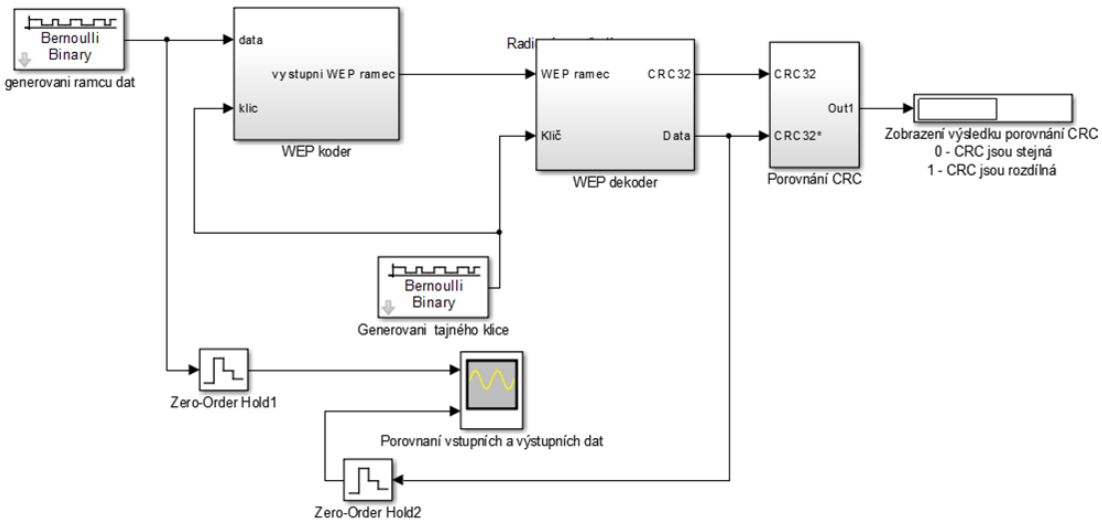
5.2 WEP

Protože ne všechny bloky jsou v Matlabu obsaženy, bylo nutné vytvořit vlastní bloky, které implementují požadované funkce. Takovým blokem je například proudová šifra RC4. Drobné funkční bloky například *Vložení inicializačního vektoru do rámce* jsou řešeny jednoduchou operací *for* pro spojování a oddělování dat a nebudou zde rozepisovány.

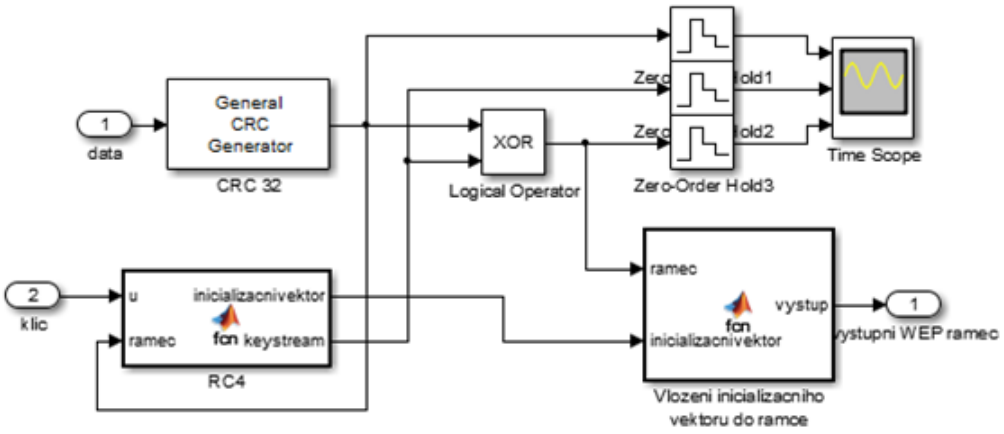
Vlastní model podporuje zpracovaní zpráv o délce více bajtů, ale v rámci přehlednosti grafů je zobrazena délka dat pouze 2 bajty.

5.2.1 Celkový model

Z modelu je vidět, že pro generování dat a klíče jsou použity náhodné generátory. Bloky WEP kodér a dekodér obsahují subsystémy, které provádějí požadovanou funkci. Blok porovnání CRC slouží k porovnání CRC32 hodnot. K tomu je použito jednoduchého řešení s XOR s ověřením, jestli výsledný rámec je nulový. Celkový model je na obr. 5.1



Obr. 5.1: Celkový model WEP

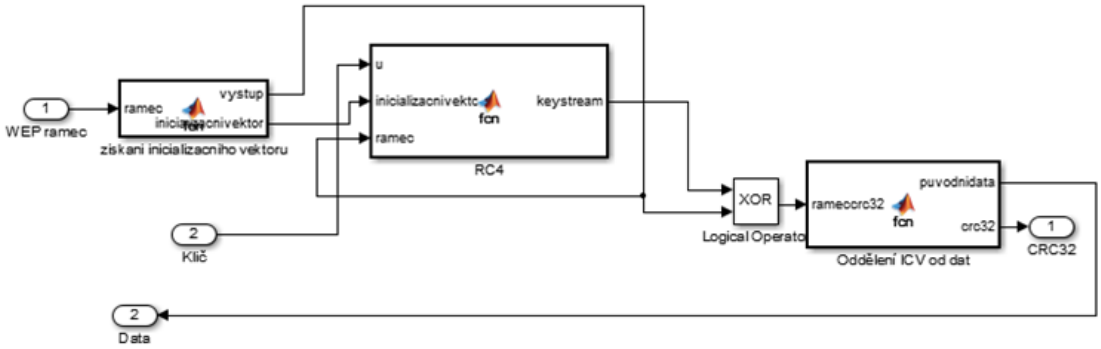


Obr. 5.2: Model WEP kodéru

5.2.2 Kodér

Model z obr. 5.2 odpovídá blokovému schématu z teorie ohledně kodéru WEP. RC4 blok v Matlabu neexistuje a proto ho bylo nutné navrhnout.

5.2.3 Dekodér



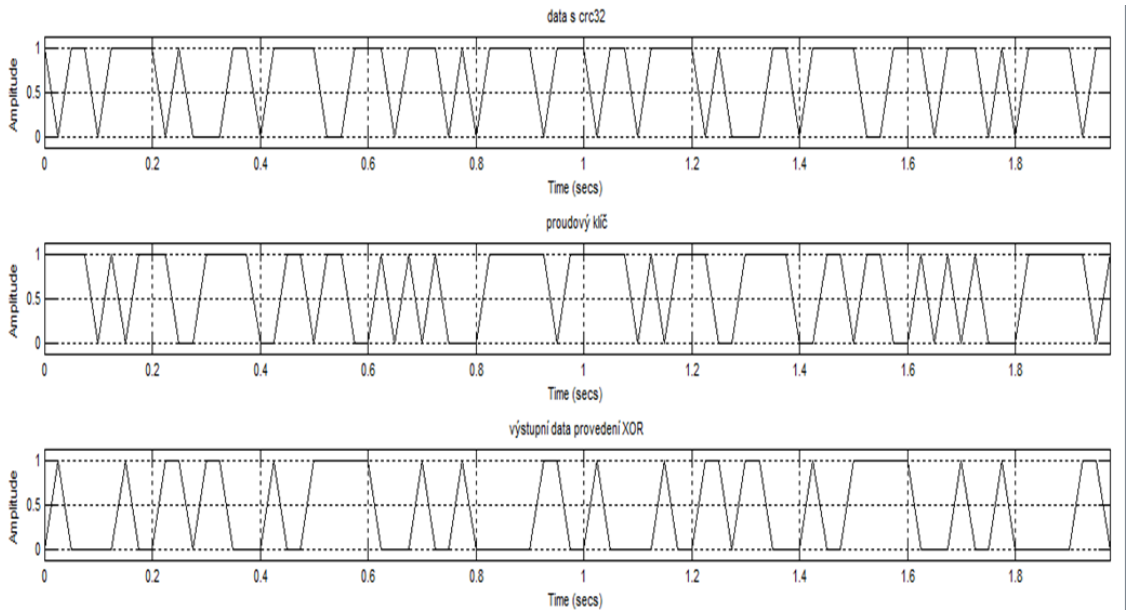
Obr. 5.3: Model WEP dekodéru

Model z obr. 5.3 odpovídá blokovému schématu z teorie ohledně dekodéru WEP. RC4 blok je v podstatě stejný jako u kodéru, jediným rozdílem je, že IV není generováno uvnitř bloku, ale je získáno ze vstupního rámce.

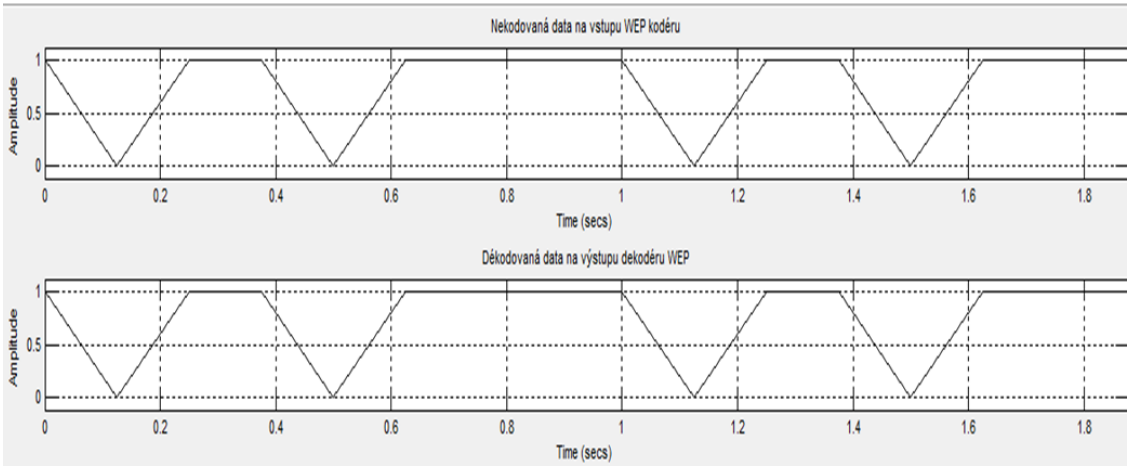
5.2.4 Blok RC4

RC4 blok bylo nutné navrhnout, protože není součástí již existujících bloků v Matlabu. Kód byl vytvářen na základě literatury ohledne KSA a PRGA algoritmů a upraven do prostředí Matlab. Vlastní kód bloku RC4 je k nalezení v příloze A.1.

5.2.5 Grafy pro WEP



Obr. 5.4: Porovnání průběhů po jednotlivých operacích. První graf zachycuje data+CRC32, druhý proudovou šifru vytvořenou na výstupu bloku RC4, třetí ukazuje výstupní data po provedení XOR operace

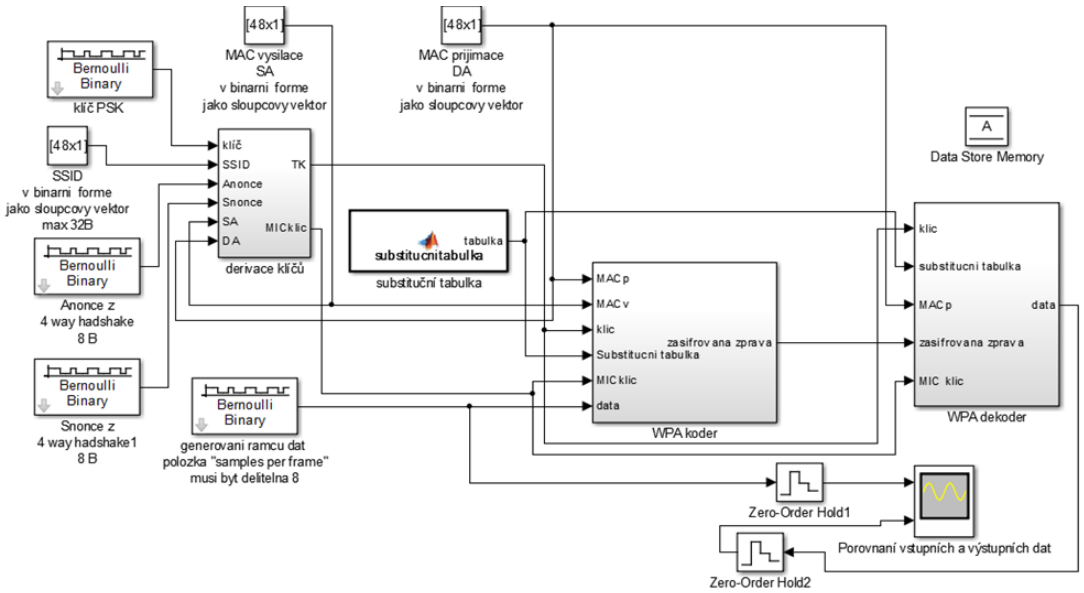


Obr. 5.5: Porovnání vstupních dat do kodéru a výstupních dat z dekodéru

5.3 WPA

Podobně jako u WEP bylo i zde nutné některé bloky navrhnout. Jako vstupní data byl zvolen krátký rámec pro snadnější orientaci v jednotlivých grafech.

5.3.1 Celkový model



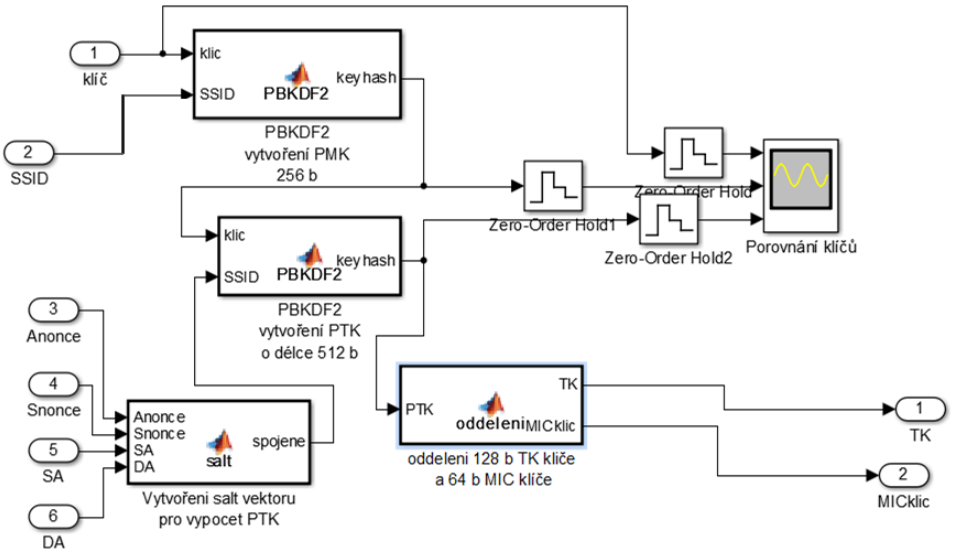
Obr. 5.6: Celkový model WPA

Vlastní model je obr. 5.6. Anonce a Snonce jsou pseudonáhodná čísla generovaná pomocí náhodného generátoru, obě mají délku 8 B. MAC vysílače a přijímače jsou MAC adresy komunikujících zařizení definovány jako konstanty o délce 6 B. Substituční tabulka je blok generující S-box, který je potřeba pro provádění substitucí v dalších blocích. Generování rámců dat je náhodný generátor s minimální délkou rámce 8 b. Klíč PSK generuje tajný klíč, pro správnou fukci musí mít klíč od 32 b do 256 b s krokem po 8 b. Blok derivace klíčů slouží k odvození jednotlivých klíčů z tajného klíče. WPA kodér a dekodér jsou bloky,které plní vlastní funkci šifrování a dešifrování dat.

5.3.2 Blok derivace klíčů

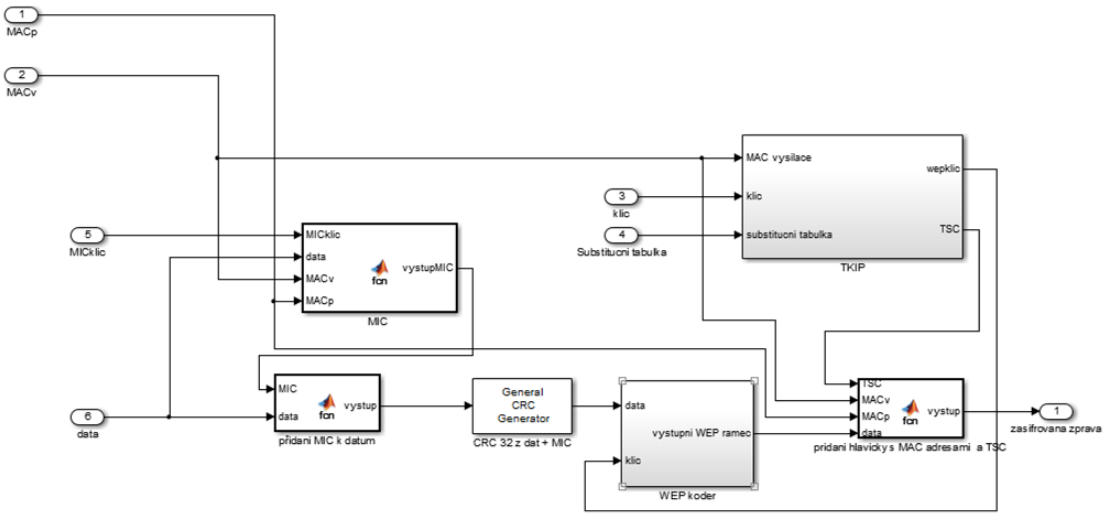
Tento blok, jehož vnitřní uspořádání je na obr. 5.7, je určen k odvození potřebných klíčů z tajného klíče. Jádrem tohoto bloku je funkce PBKDF2, kterou bylo nutné dle specifikací navrhnout. Zdrojový kód této funkce lze nalézt v příloze B.2. Vstup klíč u PBKDF2 slouží pro klíč, z kterého se odvozují další klíče, vstup SSID pro salt

vektor, který klíč ze vstupu klíč dle hashovací funkce šifruje. Keyhash je výstupní klíč po provedení operací v PBKDF2. Blok salt je určen pro sloučení vstupních informací na salt vektor pro druhý blok PBKDF2. Blok oddělení následně odděluje potřebné klíče od celkového výstupního vektoru keyhash.



Obr. 5.7: Blok derivace klíčů

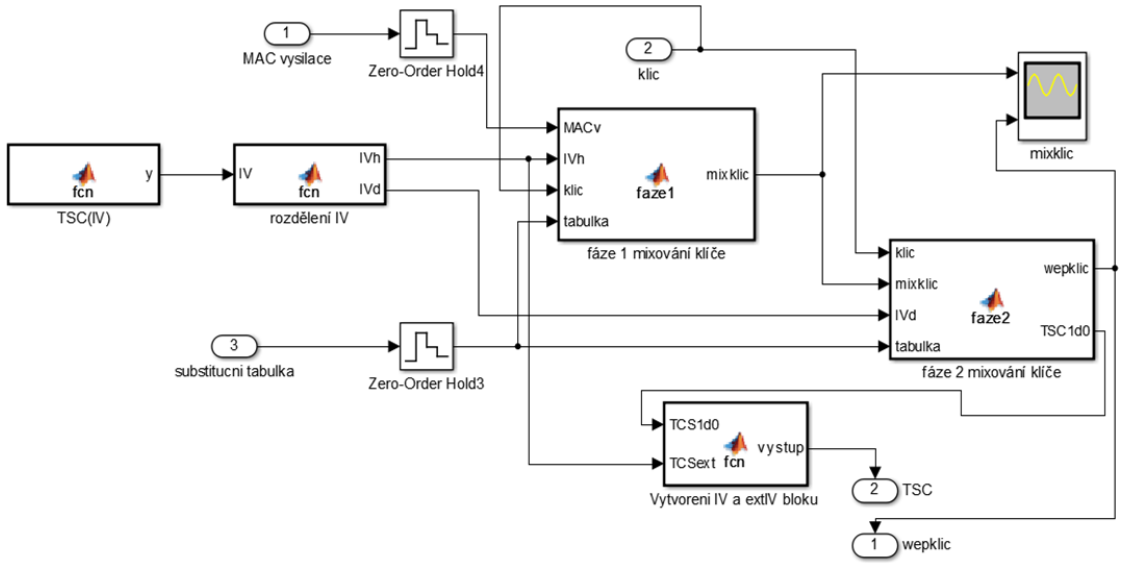
5.3.3 WPA kodér



Obr. 5.8: Blok WPA kodér

Tento blok slouží k vlastnímu šifování zpráv, jeho interní struktura je na obr. 5.8 WEP kodér je blok, který odpovídá původnímu bloku WEP kodéru u WEP šifrování na obr. 5.2. Funkce pro výpočet integrity zprávy MIC byla vytvořena na základě definic z standardů a její zdrojový kód je možné nalézt v příloze B.5. Blok TKIP slouží ke generování dočasných klíčů pro WEP kodér.

5.3.4 TKIP

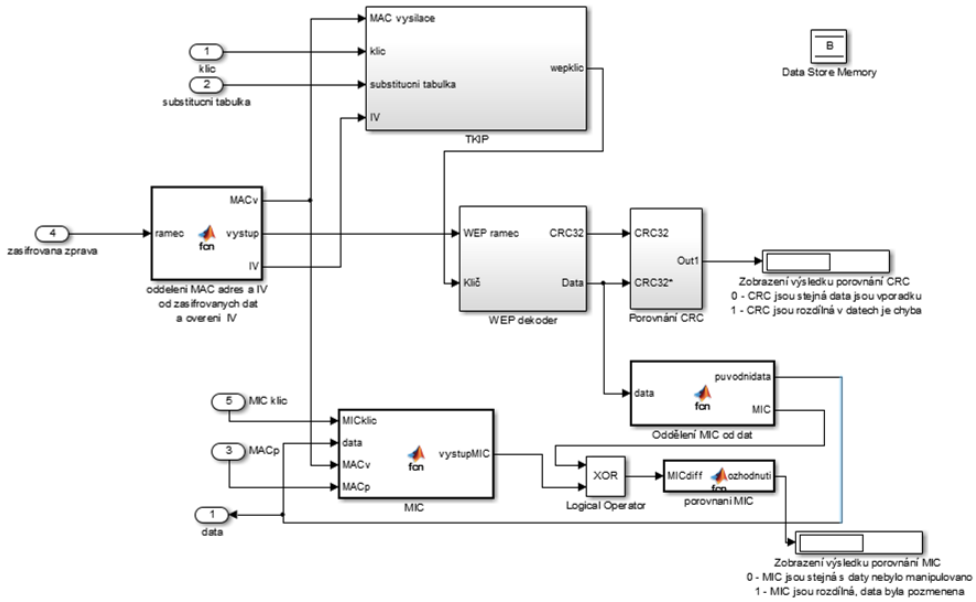


Obr. 5.9: Blok TKIP

Blok TKIP slouží k tvorbě dočasných klíčů. Vnitřní uspořádání je na obr. 5.9. Blok TSC je určen ke generování inicializačních vektorů a k jeho řešení bylo zvoleno použití globální proměnné, aby hodnota mohla být inkrementována v každém cyklu. Bloky fáze 1 mixování klíče a fáze 2 mixování klíče byly navženy dle standardu a jejich kód je k nalezení v přílohách B.3 B.4.

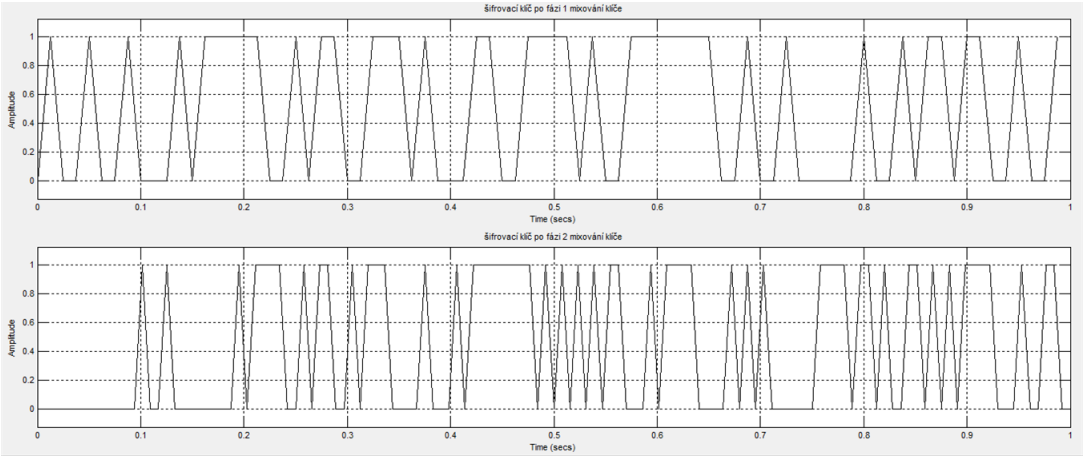
5.3.5 WPA dekodér

Tato část slouží k dešifrování příchozí zprávy. Blok TKIP je stejný jako v případě kodéru. WEP dekodér odpovída schématu na obr. 5.3. Další bloky jsou určeny k vygenerování a porovnání MIC a CRC32 s odpovídajícími hodnotami z příchozího rámce a následnému zobrazení výsledku porovnání. Vnitřní zapojení WPA dekodéru je na obr. 5.10.

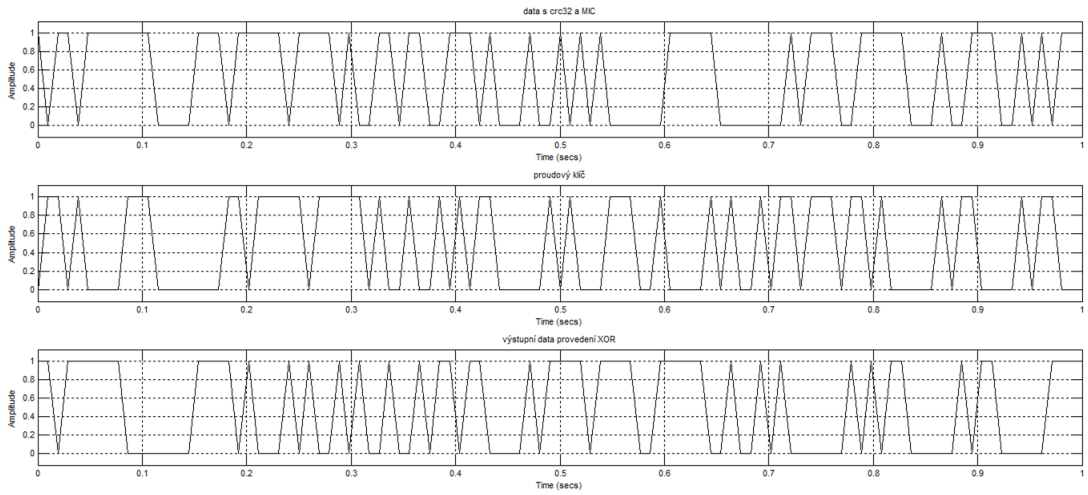


Obr. 5.10: Blok WPA dekodér

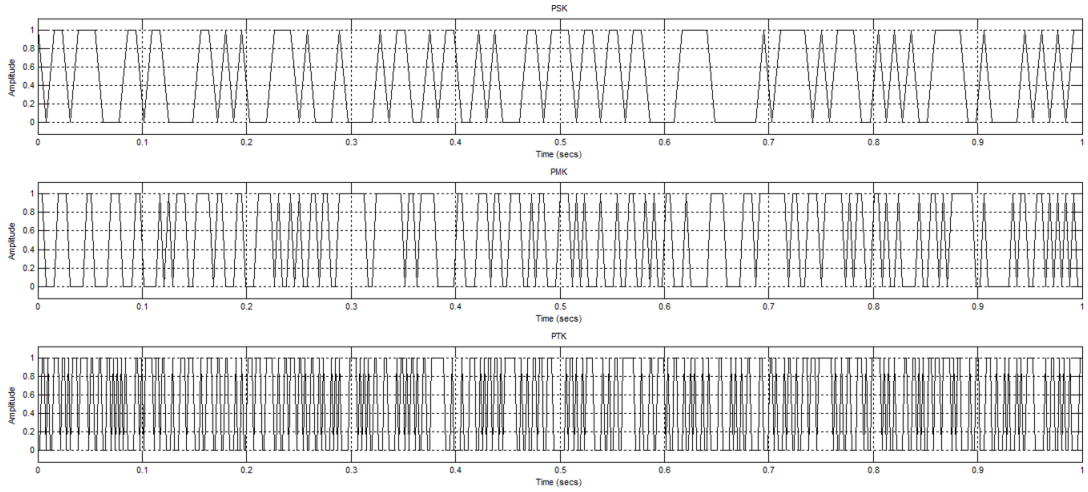
5.3.6 Grafy pro WPA



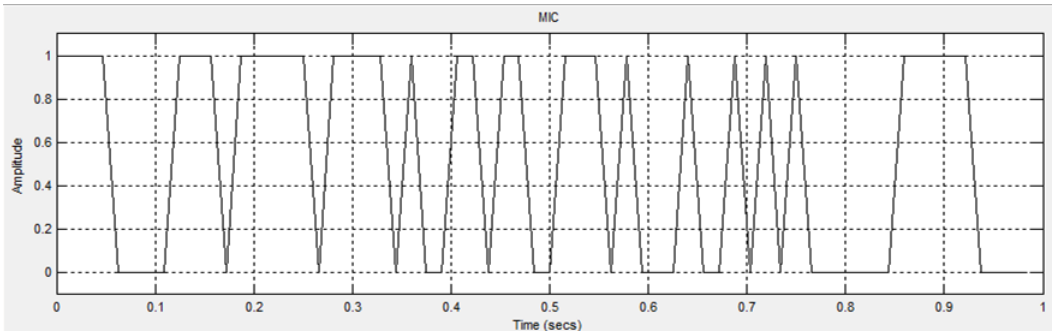
Obr. 5.11: Graf výstupů fáze 1 a 2 mixováni klíče



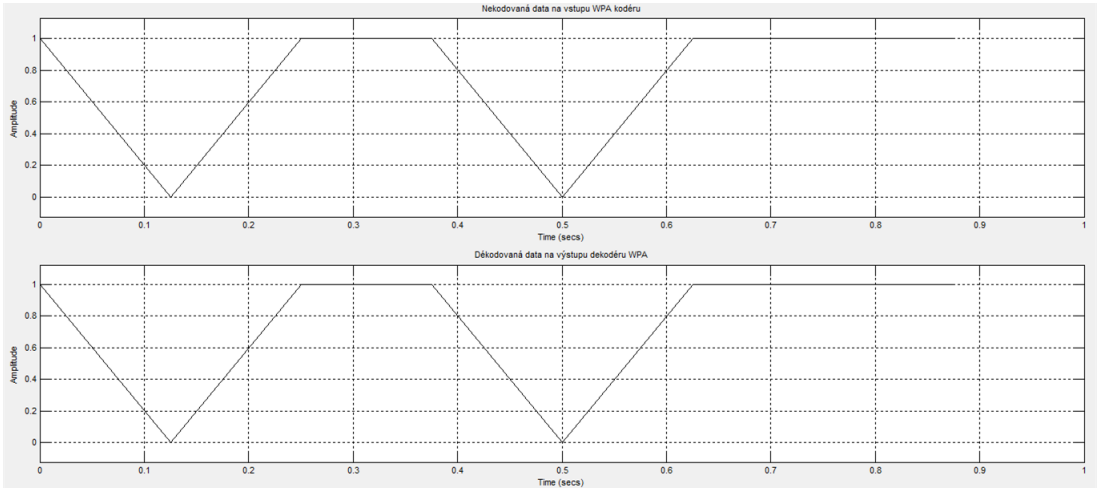
Obr. 5.12: Graf zobrazení výstupů, v prvním jsou vstupní data opatřená MIC a CRC32, v druhém proudový klíč a v třetím výstup WPA kodéru bez TSC a MAC hlavicky



Obr. 5.13: Graf zobrazení derivace klíčů, v prvním je původní klíč PSK, v druhém hashováním odvozený PMK a v třetím PTK odvozený hashováním PMK



Obr. 5.14: Výstup MIC bloku

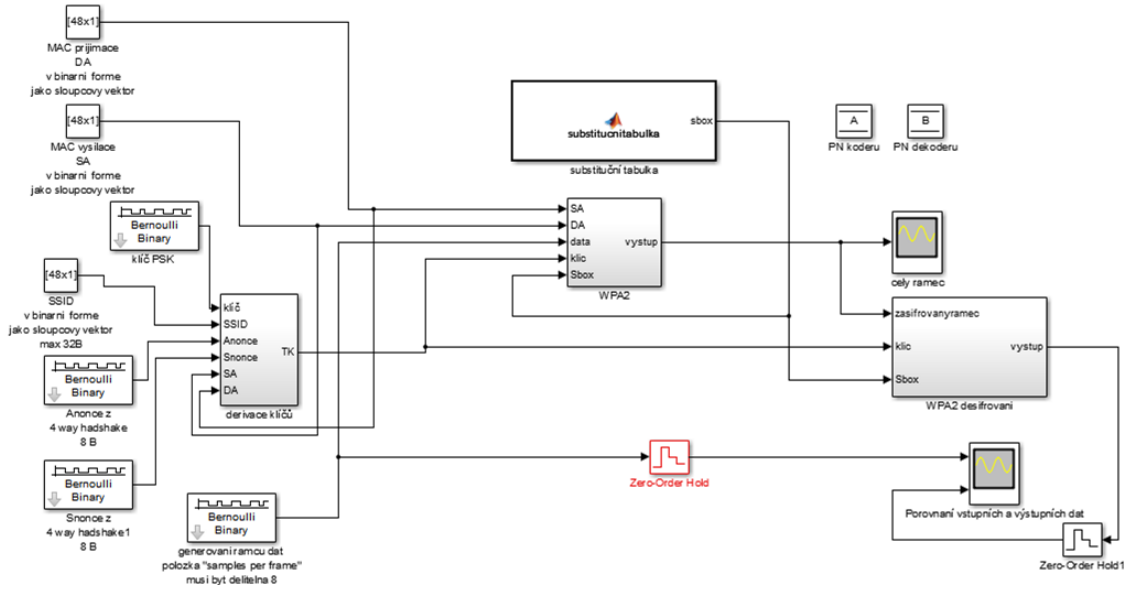


Obr. 5.15: Porovnání vstupních dat před šifrováním a výstupních dat po dešifrování

5.4 WPA2

Podobně jako u předchozích modelů, i v tomto bylo nutné navhnout vlastní bloky.

5.4.1 Celkový model WPA2

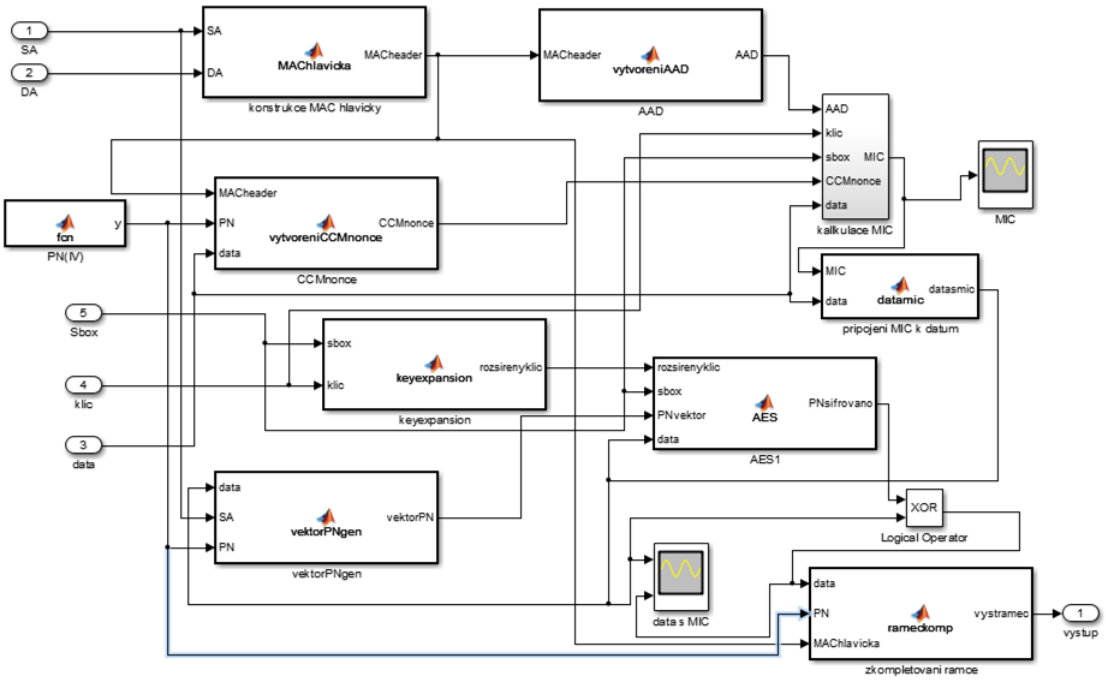


Obr. 5.16: Celkový model WPA2

Celkový model pro WPA2 je na obr. 5.16. Blok derivace klíčů je stejný jako ve WPA. Blok WPA2 realizuje šifrování a blok WPA2 dešifrování následně dešifrování dat. Substituční tabulka generuje vektor hodnot, který je určen pro substituce v dalších blocích modelu. Proměnné byly navrženy jako binarní sloupcové vektory a globální proměnné jako dekadické pro zjednodušení jejich inkrementování.

5.4.2 kodér WPA2

Tento blok slouží k vlastnímu šifrování dle WPA2 a jeho vnitřní uspořádání je na obr. 5.17. Konstrukce MAC hlavičky vytváří potřebná data pro záhlaví rámce. AAD následně z hlavičky vybere potřebná data, která mají být zabezpečena pomocí MIC. CCMnonce vytváří vektor nutný pro inicializaci MIC. Keyexpansion blok generuje rozšířený klíč pro jednotlivé bloky AES kodéru. VektorPNgen slouží k vytvoření počítadla, které je po zašifrování AES určeno k XOR operaci s vlastními daty určenými k zašifrování. Blok AES1 imlementuje šifrovací algoritmus AES pro šifrování ve WPA2. Blok zkompletování rámce je určen k přiložení všech potřebných dat



Obr. 5.17: Kodér WPA2

jako záhlaví k šifrovaným datům, aby dekodér byl schopen zašifrovaná data úspěšně dešifrovat.

5.4.3 Substituční tabulka

V tomto bloku se vypočítávají jednotlivé hodnoty pro substituce realizované v dalších blocích. Zdrojový kód substiční tabulky je v příloze C.1. V praxi se vlastní výpočet tabulky neprovádí, protože tabulka je vytvořena jako záznam v paměti zařízení.

5.4.4 Keyexpansion

Realizuje expanzi klíče na lineární pole, kdy toto pole se používá pro AES bloky v modelu jako šifrovací klíč. Zdrojový kód tohoto bloku je v příloze C.2.

5.4.5 AES

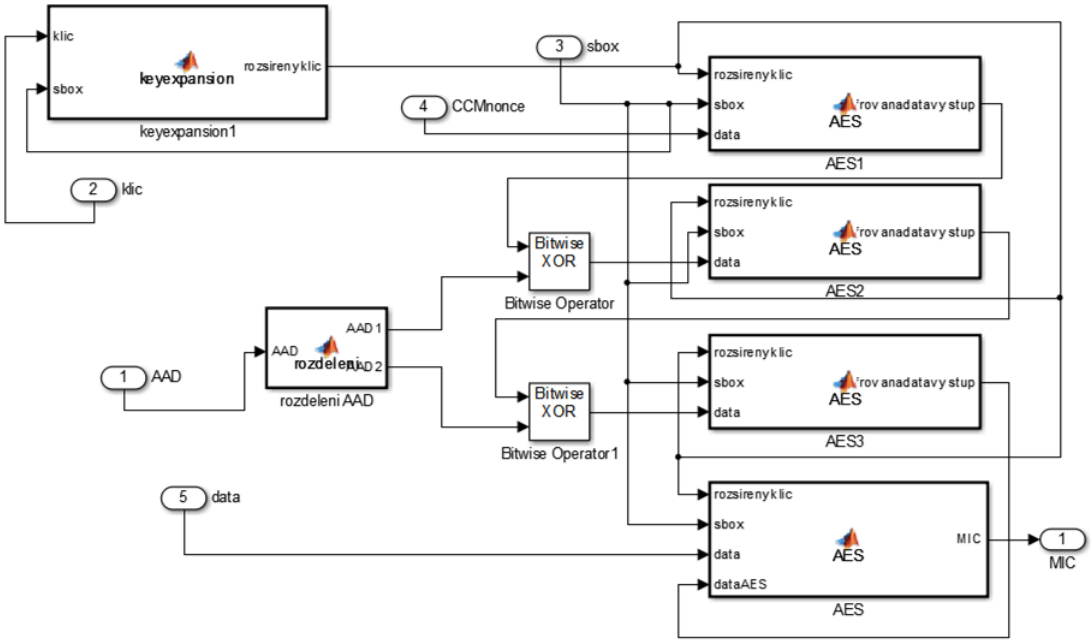
AES je šifrovací algoritmus, který je jádrem celého WPA2 zabezpečení. Jeho realizace v Matlabu byla výhodná, protože vnitřní operace tohoto bloku jsou na bázi maticového počtu. Zdrojový kód pro AES blok se nachází v příloze C.4.

5.4.6 VektorPNgen

Slouží ke generování vektoru počítadla, které musí mít odpovídající délku jako šifrovaná data. Zdrojový kód je v příloze C.3.

5.4.7 MIC

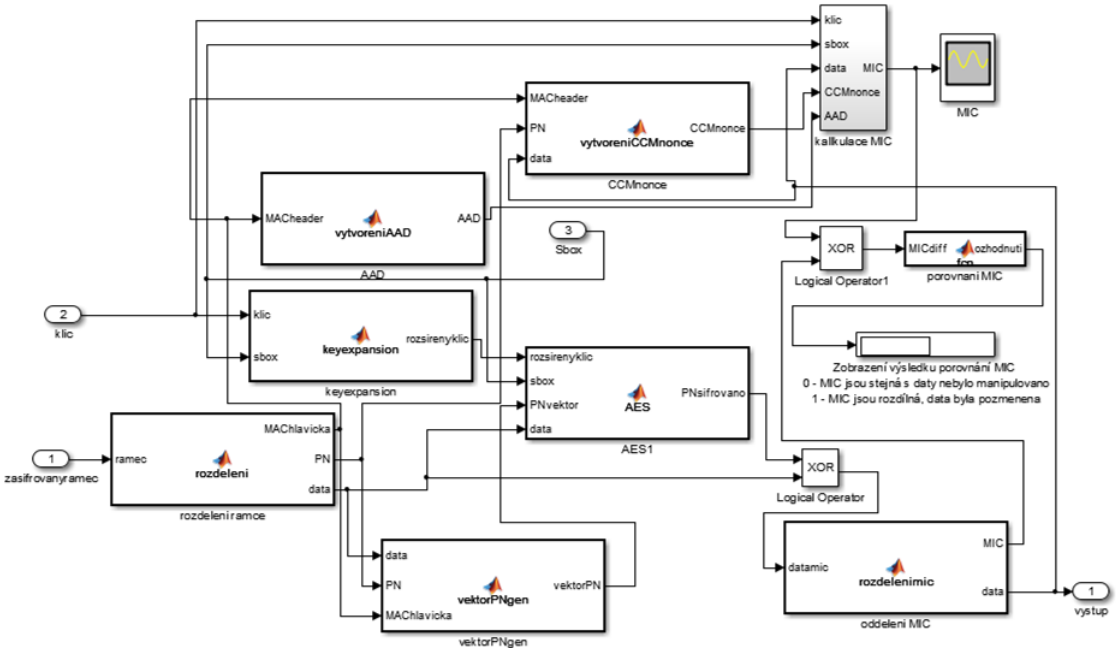
Blok, který počítá MIC hodnotu pro daný vstup. Vnitřní struktura MIC je na obr. 5.18. AES1,AES2 a AES3 slouží k šifrování záhlaví rámce. Blok AES následně pro šifrování dat, kdy konečným výsledkem je 64 b MIC.



Obr. 5.18: Blok MIC ve WPA2

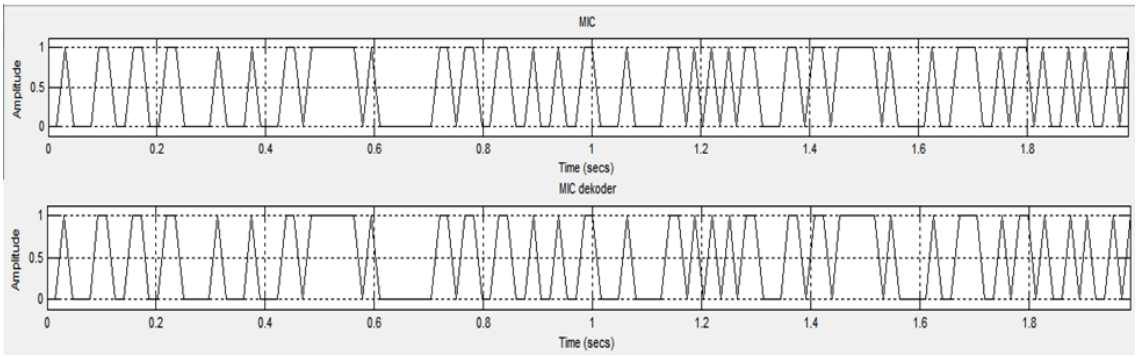
5.4.8 Dekodér WPA2

Bloky tvořící dekodér jsou odpovídají blokům v kodéru. Rozdíl je pouze v datech ze záhlaví rámce, které dekodér získá z příchozího rámce a díky kterým je schopen zprávu dešifrovat a v blocích ověřujících pravost přijatého rámce. Vnitřní struktura WPA2 dekodéru je na obr. 5.19.

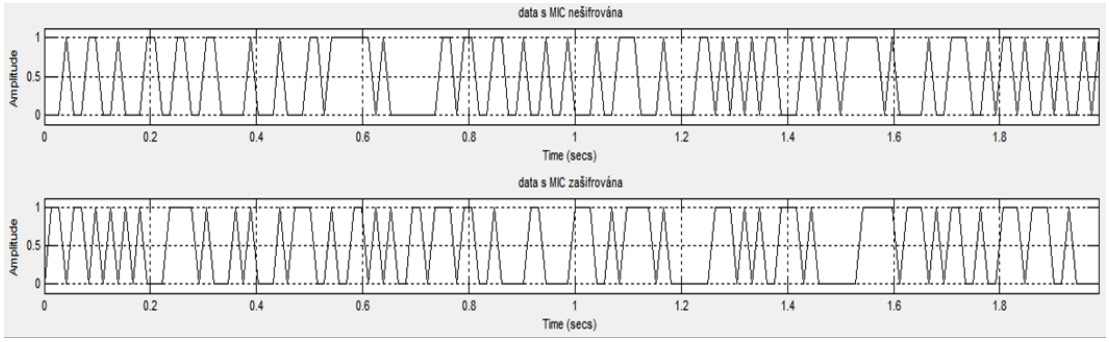


Obr. 5.19: Dekodér WPA2

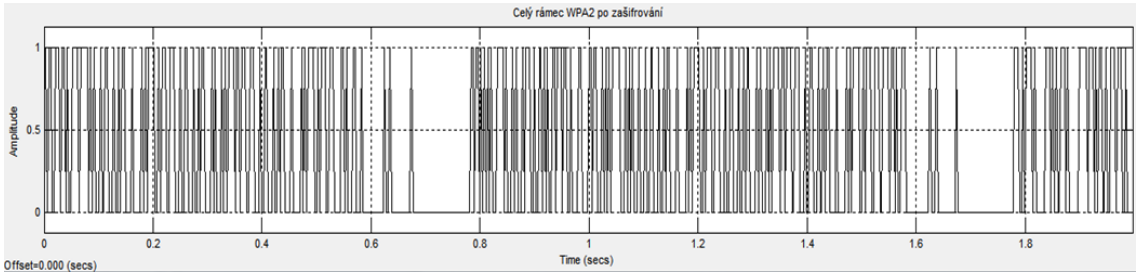
5.4.9 Grafy pro WPA2



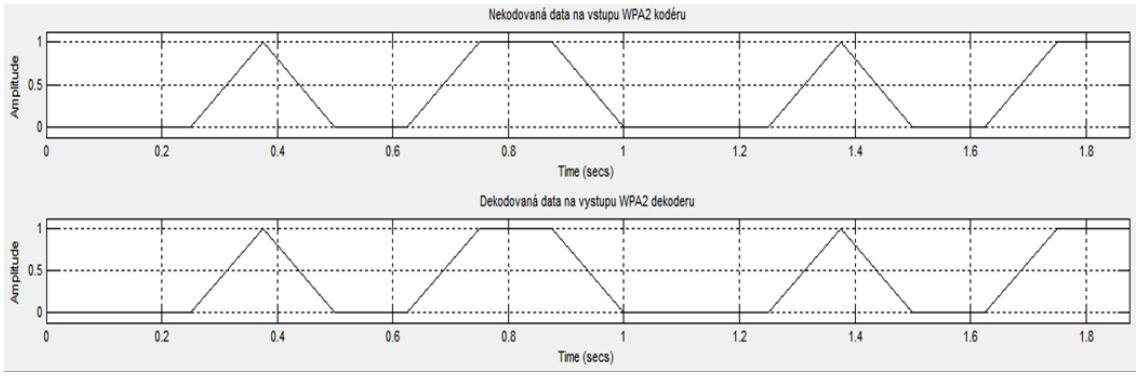
Obr. 5.20: Graf zobrazující vypočítané MIC v kodéru s dekodéru



Obr. 5.21: Porovnání nešifrovaných dat s MIC a šifrovaných dat s MIC



Obr. 5.22: Celkový rámec se zašifrovanými daty



Obr. 5.23: Porovnání vstupních nešifrovaných dat s výstupními dešifrovanými daty

6 ZÁVĚR

V této práci byly rozebrány jednotlivé metody šifrování v 802.11. U každého algoritmu byl popsán systém, jakým probíhá zabezpečení rámce dat. To bylo doplněno o bloková schémata, která tento postup ukazují. U jednotlivých algoritmů byla rozebrána slabá místa a tedy možnosti napadnutelnosti jednotlivých algoritmů.

Dle rozboru bylo ukázáno, že algoritmus WEP má mnoho slabých míst a je tedy snadno prolomitelný. Proto platí, že v současnosti není doporučeno používat zabezpečení WEP.

WPA, i když je postaveno na WEP, se ukazuje jako odolné proti útokům. Platí, že pokud je použit kvalitní klíč, tak „bruteforce“ útok by trval příliš dlouho a i když existuje „Beck and Tews“ útok, tak tento útok nedokáže zjistit šifrovací klíč a nejedná se tedy o zásadní prolomení zabezpečení.

WPA2 se dle teorie ukázalo jako nejbezpečnější řešení. I zde platilo, že pokud je použit slabý klíč, tak existuje možnost prolomit toto zabezpečení. Nevýhodou tohoto algoritmu jsou díky jeho složitosti vyšší nároky na výkon zařízení.

V rámci praktické části byly realizovány modely WEP, WPA a WPA2 v prostředí Matlab simulink. K tomu bylo nutné vytvořit nové bloky, protože bloky obsažené v simulinku byly pro potřeby realizace nedostatečné. Tyto bloky byly napsány jako kód do funkčního bloku v simulinku. Jejich návrh se řídil dle definic daných standardy pro jednotlivé typy zabezpečení. Po zprovoznění bylo ověřeno, že data na vstupu odpovídají datům dešifrovaných na výstupu dekodéru dle typu realizovaného zabezpečení. V modelech byly zobrazeny grafy průběhů na výstupech některých vnitřních bloků.

V průběhu řešení modelování zabezpečení se vyskytl i problém s použitím ASCII znaků pro tajné heslo a hexadecimálních čísel pro MAC adresy. Bohužel prostředí simulink signály s obsahem jiným než čísla nepodporuje, proto hodnoty těchto signálů jsou převedeny na binární formát. Převedení na binární formát bylo provedeno pomocí příkazu hex2bin pro hexadecimální číslo a dec2bin pro znak ASCII v příkazové řádce matlabu.

LITERATURA

[1] *802.11 security. wi-fi protected access and 802.11i.* ETutorials.org [online]. 2008-2014 [cit. 2014-05-24]. Dostupné z: <http://etutorials.org/Networking/802.11+security.+wi-fi+protected+access+and+802.11i/>.

[2] *Authentication Types for Wireless Devices.* CISCO. [online]. © 1992-2013 [cit. 2013-12-29]. Dostupné z: <http://www.cisco.com/en/US/docs/routers/access/wireless/software/guide/SecurityAuthenticationTypes.html>.

[3] ARKASHA a BOBZILLA. *WiGLE: Wireless Geographic Logging Engine.* WiGLE.net [online]. Copyright 2001-2014 [cit. 2014-05-24]. Dostupné z: <https://wigle.net/gps/gps/main/ssidstats>.

[4] *Cisco Unified Wireless Network Architecture—Base Security Features.* CISCO. [online]. © 1992-2013 [cit. 2013-12-29]. Dostupné z: <http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob41dg/ch4_Secu.html>.

[5] EASTLAKE, D. MOTOROLA. rfc3174. *US Secure Hash Algorithm 1.* Network Working Group, September 2001 [cit. 2014-5-19]. Dostupné z: <http://www.rfc-base.org/txt/rfc-3174.txt>.

[6] HALVORSEN, Finn Michael a Olav HAUGEN. *Cryptanalysis of IEEE 802.11i TKIP* [online]. Trondheim, June 2009 [cit. 2013-12-29]. Dostupné z: <http://hirte.aircrack-ng.org/tkip_master.pdf>. Master of Science in Communication Technology. Norwegian University of Science and Technology. Vedoucí práce Stig Frode Mjølsnes, ITEM.

[7] CHANDRA, Praphul. *Bulletproof wireless security: GSM, UMTS, 802.11 and ad hoc security.* Boston: Newnes, 2005, xxxiii, 237 p. ISBN 07-506-7746-5.

[8] IEEE Std 802.11™-2007. *IEEE Standard for Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* 3 Park Avenue New York, NY 10016-5997, USA: IEEE, 2007. Dostupné z: <http://standards.ieee.org/about/get/802/802.11.html>.

[9] IEEE Std 802.11i/D2.5. *Draft Supplement to STANDARD FOR Telecommunications and Information Exchange Between Systems.* 345 East 47th Street New York, NY 10017, USA: Institute of Electrical and Electronics Engineers, Inc., 2002.

[10] KAK, Avi. *Lecture 8: AES: The Advanced Encryption Standard.* Purdue University, 2014. Dostupné z: <https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf>. Lecture Notes. Purdue University.

[11] KALISKI, B. RSA LABORATORIES. rfc2898. *PKCS #5: Password-Based Cryptography Specification Version 2.0.* Network Working Group, September 2000 [cit. 2014-5-19]. Dostupné z: <http://tools.ietf.org/html/rfc2898"> http://tools.ietf.org/html/rfc2898>.

[12] KRAWCZYK, H. IBM. rfc2104. *HMAC: Keyed-Hashing for Message Authentication.* Network Working Group, February 1997 [cit. 2014-5-19]. Dostupné z: <http://tools.ietf.org/html/rfc2104>

[13] KOCUR, Zbyněk. *Bezdrátové sítě dle standardu IEEE 802.11 (WiFi)* [online]. [2006] [cit. 2013-12-29]. Dostupné z: <http://www.comtel.cz/files/download.php?id=2229>.

[14] LEHEMBRE, Guillaume. *Wi-Fi security – WEP, WPA and WPA2.* Hackin9 [online]. 2005 [cit. 2013-12-29]. Dostupné z: <https://www.msec.be/vincent/sec/background/WIFISECpaper.pdf>.

[15] MATHWORKS. *Simulink Documentation Center* [online]. © 1994-2013 [cit. 2013-12-29]. Dostupné z: <http://www.mathworks.com/help/simulink/>.

[16] POOLE, Ian. *IEEE 802.11 standards tutorial.* © ADRIO COMMUNICATIONS LTD. Radio-Electronics.com: Resources and analysis for electronics engineers [online]. 13.1.2010 [cit. 2013-12-28]. Dostupné z: <http://www.radio-electronics.com/info/wireless/wi-fi/ieee-802-11-standards-tutorial.php>.

[17] *SHA-1* [online], poslední aktualizace 14 Března 2014 02:24 [cit. 2014-5-19], Wikipedie. Dostupné z WWW: <http://en.wikipedia.org/wiki/SHA-1>.

[18] *Technical Overview of IEEE 802.11 WLAN Standards.* Agilent technologies [online]. 3.9.2012 [cit. 2013-12-29]. Dostupné z: <http://www.agilent.com/about/newsroom/tmnews/background/WLAN/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AAD rozšířená autentizační data – Additional Authentication Data

AES pokročilý šifrovací standard – Advanced Encryption Standard

AP přístupový bod – Access Point

CCMP Counter Mode with Cipher Block Chaining Message Authencation Code Protocol

CKK doplňkové kódové klíčování – Complementary Code Keying

CRC kontrolní součet – Cyclic Redundancy Check

CTR počítadlo – Counter

DSSS technika přímého rozprostírání spektra – Direct Sequence Spread Spectrum

EAP autentizační protokol zprostředkující přenos klíčů – Extensible Authentication Protocol

FHSS technika rozprostírání spektra s přeskakováním frekvence nosné – Frequency Hopping Spread Spectrum

GTK skupinový dočasný klíč – Group Temporal Key

HMAC funkce generování autentizační zprávy pomocí hash funkce – Hash-based Message Authentication Code

IEEE institut sdružující odborníky v oboru elektrické a elektrotechnické standardizace – The Institute of Electrical and Electronics Engineers

IV inicializační vektor – Inicialisation Vector

KSA algoritmus rozvrhnutí klíče – Key Scheduling Algorithm

MAC identifikátor síťového zařízení – Media Access Control

MIC kontrola integrity zpráv – Message Integrity Check

MIMO metoda vícenásobného vstupu a výstupu – Multiple Input Multiple Output

OFDM ortogonální multiplex s kmitočtovým dělením – Orthogonal Frequency Division Multiplex

PBKDF2 pseudonáhodný generátor – Pseudo-Random Generation Algorithm

PMK párový primární klíč – Pairwise Master Key

PN číslo paketu – Packet Number

PRGA funkce derivace klíče na základě hesla – Password-Based Key Derivation Function 2

PSK předsdílený klíč – Pre-Shared Key

PTK párový přenosový klíč – Pairwise Transient Key

QoS kvalita služeb – Quality of Service

SHA1 bezpečný hashovací algoritmus – US Secure Hash Algorithm 1

TK dočasný klíč – Temporal Key

TKIP protokol dočasné integrity klíče – Temporal Key Integrity Protocol

TSC TKIP sekvenční počítadlo –TKIP Sequence Counter

WEP soukromí ekvivalentní drátovým sítím – Wired Equivalent Privacy

WPA chráněný přístup k Wi-Fi – Wi-Fi Protected Access

WPS chráněné nastavení Wi-Fi – Wi-Fi Protected Setup

SEZNAM PŘÍLOH

A ZDROJOVÉ KÓDY PRO WEP

A.1 Zdrojový kód bloku RC4

```
function [inicializacnivektor,keystream] = fcn(u,ramec)
IV = rand(24,1)<0.5;     %vytvoreni inicializacniho vektoru
y = [u;IV];           %vytvoreni 64bitoveho klice
coder.extrinsic('vec2mat', 'bi2de','de2bi');
% povoleni pouziti externich prikazu z matlabu
y1=zeros(8,8,'int32');   % inicializace pro coder.extrinsic
y1=vec2mat(y,8);   % timhle to prevedu matici kde je 8 bitu v radku
key=zeros(8,1,'int32');  % inicializace pro coder.extrinsic
key = bi2de(y1); % prevadi 8b v klici na dekadicke sloupcovy vektor
len = length(y)/8;   % delka klice v bajtech
S=zeros(256,1);
pom=zeros(8,1,'int32');

%KSA generovani subklice

for i = 1:1:256      % vytvoreni vectoru s hodnotami 0 az 255
    S(i) = i-1;
end
 j = int32(1);
for i = 1:1:256     %promichani hodnot ve vektoru S

    j = mod((j-1+S(i)+key((mod(i-1,len)+1))), 256);
    j=j+1;
    pomocna = S(i);
    S(i)=S(j);
    S(j)=pomocna;
end

%PRGA generovani vystupniho keystreamu
i = int32(0);
j = int32(0);
out =int32(zeros(256,1));
for z= 1:1:length(ramec)
    i = mod((i + 1),256);
```

```
      j = mod((j + S(i+1)),256);
       pomocna = S(i+1);
        S(i+1)=S(j+1);
        S(j+1)=pomocna;
      out(z) = S(mod((S(i+1) + S(j+1)),256)+1);

end
      %prevedeni zpet na tok bitu
    keystream=zeros(length(ramec),1);
for zz= 1:1:length(ramec)/8    % prevod kazdeho byte na bity
    pom=de2bi(out(zz),8);
    for zk = 1:1:8 % zapsani vlastnich bitu do keystreamu
        keystream(((zz-1)*8)+zk)=pom(zk);
    end
end

inicializacnivektor = IV;
% poslani inicializacniho vektoru na vystup bloku
```

B WPA

B.1 WPA S-Box dle standardu

```
S-box lower byte
{0xA5,0x84,0x99,0x8D,0x0D,0xBD,0xB1,0x54,
0x50,0x03,0xA9,0x7D,0x19,0x62,0xE6,0x9A,
0x45,0x9D,0x40,0x87,0x15,0xEB,0xC9,0x0B,
0xEC,0x67,0xFD,0xEA,0xBF,0xF7,0x96,0x5B,
0xC2,0x1C,0xAE,0x6A,0x5A,0x41,0x02,0x4F,
0x5C,0xF4,0x34,0x08,0x93,0x73,0x53,0x3F,
0x0C,0x52,0x65,0x5E,0x28,0xA1,0x0F,0xB5,
0x09,0x36,0x9B,0x3D,0x26,0x69,0xCD,0x9F,
0x1B,0x9E,0x74,0x2E,0x2D,0xB2,0xEE,0xFB,
0xF6,0x4D,0x61,0xCE,0x7B,0x3E,0x71,0x97,
0xF5,0x68,0x00,0x2C,0x60,0x1F,0xC8,0xED,
0xBE,0x46,0xD9,0x4B,0xDE,0xD4,0xE8,0x4A,
0x6B,0x2A,0xE5,0x16,0xC5,0xD7,0x55,0x94,
0xCF,0x10,0x06,0x81,0xF0,0x44,0xBA,0xE3,
0xF3,0xFE,0xC0,0x8A,0xAD,0xBC,0x48,0x04,
0xDF,0xC1,0x75,0x63,0x30,0x1A,0x0E,0x6D,
0x4C,0x14,0x35,0x2F,0xE1,0xA2,0xCC,0x39,
0x57,0xF2,0x82,0x47,0xAC,0xE7,0x2B,0x95,
0xA0,0x98,0xD1,0x7F,0x66,0x7E,0xAB,0x83,
0xCA,0x29,0xD3,0x3C,0x79,0xE2,0x1D,0x76,
0x3B,0x56,0x4E,0x1E,0xDB,0x0A,0x6C,0xE4,
0x5D,0x6E,0xEF,0xA6,0xA8,0xA4,0x37,0x8B,
0x32,0x43,0x59,0xB7,0x8C,0x64,0xD2,0xE0,
0xB4,0xFA,0x07,0x25,0xAF,0x8E,0xE9,0x18,
0xD5,0x88,0x6F,0x72,0x24,0xF1,0xC7,0x51,
0x23,0x7C,0x9C,0x21,0xDD,0xDC,0x86,0x85,
0x90,0x42,0xC4,0xAA,0xD8,0x05,0x01,0x12,
0xA3,0x5F,0xF9,0xD0,0x91,0x58,0x27,0xB9,
0x38,0x13,0xB3,0x33,0xBB,0x70,0x89,0xA7,
0xB6,0x22,0x92,0x20,0x49,0xFF,0x78,0x7A,
0x8F,0xF8,0x80,0x17,0xDA,0x31,0xC6,0xB8,
0xC3,0xB0,0x77,0x11,0xCB,0xFC,0xD6,0x3A}
```

```
S-box upper byte
{0xC6,0xF8,0xEE,0xF6,0xFF,0xD6,0xDE,0x91,
0x60,0x02,0xCE,0x56,0xE7,0xB5,0x4D,0xEC,
0x8F,0x1F,0x89,0xFA,0xEF,0xB2,0x8E,0xFB,
0x41,0xB3,0x5F,0x45,0x23,0x53,0xE4,0x9B,
0x75,0xE1,0x3D,0x4C,0x6C,0x7E,0xF5,0x83,
0x68,0x51,0xD1,0xF9,0xE2,0xAB,0x62,0x2A,
0x08,0x95,0x46,0x9D,0x30,0x37,0x0A,0x2F,
0x0E,0x24,0x1B,0xDF,0xCD,0x4E,0x7F,0xEA,
0x12,0x1D,0x58,0x34,0x36,0xDC,0xB4,0x5B,
0xA4,0x76,0xB7,0x7D,0x52,0xDD,0x5E,0x13,
0xA6,0xB9,0x00,0xC1,0x40,0xE3,0x79,0xB6,
0xD4,0x8D,0x67,0x72,0x94,0x98,0xB0,0x85,
0xBB,0xC5,0x4F,0xED,0x86,0x9A,0x66,0x11,
0x8A,0xE9,0x04,0xFE,0xA0,0x78,0x25,0x4B,
0xA2,0x5D,0x80,0x05,0x3F,0x21,0x70,0xF1,
0x63,0x77,0xAF,0x42,0x20,0xE5,0xFD,0xBF,
0x81,0x18,0x26,0xC3,0xBE,0x35,0x88,0x2E,
0x93,0x55,0xFC,0x7A,0xC8,0xBA,0x32,0xE6,
0xC0,0x19,0x9E,0xA3,0x44,0x54,0x3B,0x0B,
0x8C,0xC7,0x6B,0x28,0xA7,0xBC,0x16,0xAD,
0xDB,0x64,0x74,0x14,0x92,0x0C,0x48,0xB8,
0x9F,0xBD,0x43,0xC4,0x39,0x31,0xD3,0xF2,
0xD5,0x8B,0x6E,0xDA,0x01,0xB1,0x9C,0x49,
0xD8,0xAC,0xF3,0xCF,0xCA,0xF4,0x47,0x10,
0x6F,0xF0,0x4A,0x5C,0x38,0x57,0x73,0x97,
0xCB,0xA1,0xE8,0x3E,0x96,0x61,0x0D,0x0F,
0xE0,0x7C,0x71,0xCC,0x90,0x06,0xF7,0x1C,
0xC2,0x6A,0xAE,0x69,0x17,0x99,0x3A,0x27,
0xD9,0xEB,0x2B,0x22,0xD2,0xA9,0x07,0x33,
0x2D,0x3C,0x15,0xC9,0x87,0xAA,0x50,0xA5,
0x03,0x59,0x09,0x1A,0x65,0xD7,0x84,0xD0,
0x82,0x29,0x5A,0x1E,0x7B,0xA8,0x6D,0x2C}
```

B.2 Zdrojový kód PBKDF2

```
function keyhash   = PBKDF2(klic,SSID)
coder.extrinsic('padarray','de2bi');
delkaSSID = length(SSID)/8;
if (delkaSSID>32)
    error('Delka SSID prekrocila maximalni delku 32 B');
end
if (length(klic)/8>32)
    error('Delka klice prekrocila maximalni delku 32 B');
end
dklen = 32; %delka vystupu v bajtech
cycle = 4096; %pocet cyklu
U = zeros(cycle,160);
a = zeros(8,1);
subhash = zeros((ceil(dklen*8/160))*160,1);
for i = 0:ceil(dklen*8/160)-1
    init = zeros(1,160);
    a = de2bi(i,8);
    %vypocitani hashe pro kazdy cyklus tedy 4096
U(1,:) = (hmacsha1(klic,[SSID;a]))';
    for c = 2:cycle
        U(c,:) = (hmacsha1(klic,(U(c-1,:))'))';
    end
    % provedeni XOR pro kazdy cyklus
    for c = 1:cycle
        init(1,:) = xor(init(1,:),U(c,:));
    end
    subhash(i*160+1:i*160+160,1)=init(1,:)';
end
%finalni vystup
keyhash = subhash(1:dklen*8);


function hash   = hmacsha1(klic,SSID)
delkabloku = 64; % delka hashovaciho klice v bajtech SHA1
delkaSSID = length(SSID)/8;
coder.extrinsic('padarray');
%padding vektory a jejich rozsireni na 64B
```

```
ipad = [0 0 1 1 0 1 1 0]';
opad = [0 1 0 1 1 1 0 0]';
ipad64 = zeros(64*8,1);
opad64 = zeros(64*8,1);
for i = 0:8:63*8
    for j = 1:1:8
        ipad64(i+j) =ipad(j);
        opad64(i+j) =opad(j);
    end
end

a = zeros(64*8,1);
%rozsireni na 64B
if (delkaSSID < delkabloku)
    a=padarray(SSID,[(delkabloku-delkaSSID)*8 0],'post');
end

ipadkey=xor(a,ipad64);
opadkey=xor(a,opad64);
hash = SHA1(opadkey,SHA1(ipadkey,klic));


%pomocna funkce pro vypocet SHA1
function hashsha = SHA1(hashkey,hashovane)
w = zeros(80,32);

if (length(hashovane) < 512)
  pad = (de2bi(hex2dec('80'),8))';
  hashovane512 = [hashovane;pad;zeros((512-length(hashovane)-8),1)];
end
%konstanty definovane v FIPS
h0 = (de2bi(hex2dec('67452301'),32))';
h1 = (de2bi(hex2dec('EFCDAB89'),32))';
h2 = (de2bi(hex2dec('98BADCFE'),32))';
h3 = (de2bi(hex2dec('10325476'),32))';
h4 = (de2bi(hex2dec('C3D2E1F0'),32))';
vstup = [hashkey;hashovane512];
for bloky = 0:1
    for slova = 0:1:15
```

```
        w(slova+1,:)=vstup((bloky*512+slova*32+1):bloky*512+
(slova+1)*32);
    end
    for slova =  16:1:79
     w(slova+1,:) = circshift((xor(xor(w(slova-2,:),w(slova-7,:)),
 xor(w(slova-13,:),w(slova-15,:)))),-1);
    end
    a = logical(h0);
    b = logical(h1);
    c = logical(h2);
    d = logical(h3);
    e = logical(h4);
    f = logical(zeros(32,1));
    k = logical(zeros(32,1));

    %hlavni smycka
    for i=1:80
      if 1 <= i <= 20
        f = or(and(a,c),and(not(b),d));
        k = logical((de2bi(hex2dec('5A827999'),32))');
         else if 21 <= i <= 40
               f = xor(b,xor(c,d));
               k = logical((de2bi(hex2dec('6ED9EBA1'),32))');
                else if 41 <= i <= 60
f = or(or(and(b,c),and(b,d)),and(c,d));
k = logical((de2bi(hex2dec('8F1BBCDC'),32))');
                      else if 61 <= i <= 80
f = xor(b,xor(c,d));
            k = logical((de2bi(hex2dec('CA62C1D6'),32))');
                           end
                    end
                end
        end
  abc = (de2bi(bi2de((circshift(a,-5))')+bi2de((f)')+bi2de((e)')
+bi2de((k)')+bi2de(w(i,:)),64));
        e=d;
        d=c;
        c=circshift(b,-30);
        b=a;
```

```
        pom = abc(1,1:32);
        a = logical(pom');
    end
    pom0 = (de2bi(bi2de((h0)') + bi2de((a)'),34))';
    pom1 = (de2bi(bi2de((h1)') + bi2de((b)'),34))';
    pom2 = (de2bi(bi2de((h2)') + bi2de((c)'),34))';
    pom3 = (de2bi(bi2de((h3)') + bi2de((d)'),34))';
    pom4 = (de2bi(bi2de((h4)') + bi2de((e)'),34))';
    h0 = pom0(1:32);
    h1 = pom1(1:32);
    h2 = pom2(1:32);
    h3 = pom3(1:32);
    h4 = pom4(1:32);

end
 hashsha = [h0;h1;h2;h3;h4];
```

B.3 Zdrojový kód fáze 1 mixování klíče

```
function mixklic  = faze1(MACv,IVh,klic,tabulka)
coder.extrinsic('vec2mat', 'bi2de','de2bi');
% povoleni pouziti externich prikazu z matlabu
%vytvoreni pomocnych promennych
PK0=rand(16,1)<0.5;
PK1=rand(16,1)<0.5;
PK2=rand(16,1)<0.5;
PK3=rand(16,1)<0.5;
PK4=rand(16,1)<0.5;
TA0=rand(8,1)<0.5;
TA1=rand(8,1)<0.5;
TA2=rand(8,1)<0.5;
TA3=rand(8,1)<0.5;
TA4=rand(8,1)<0.5;
TA5=rand(8,1)<0.5;
TK0=rand(8,1)<0.5;
TK1=rand(8,1)<0.5;
TK2=rand(8,1)<0.5;
TK3=rand(8,1)<0.5;
TK4=rand(8,1)<0.5;
TK5=rand(8,1)<0.5;
TK6=rand(8,1)<0.5;
TK7=rand(8,1)<0.5;
TK8=rand(8,1)<0.5;
TK9=rand(8,1)<0.5;
TK10=rand(8,1)<0.5;
TK11=rand(8,1)<0.5;
TK12=rand(8,1)<0.5;
TK13=rand(8,1)<0.5;
TK14=rand(8,1)<0.5;
TK15=rand(8,1)<0.5;

for i = 1:1:16
    PK0(i) = IVh(16+i);
    PK1(i) = IVh(i);
end
```

```
for i = 1:1:8                %naplneni pomocnych promennych
  TA0(i) = MACv(i);           %nejmene dulezity bajt
  TA1(i) = MACv(8+i);
  TA2(i) = MACv(16+i);
  TA3(i) = MACv(24+i);
  TA4(i) = MACv(32+i);
  TA5(i) = MACv(40+i);              %nejvice dulezity bajt
  TK0(i) = klic(i);    %nejmene dulezity bajt
  TK1(i) = klic(8+i);
  TK2(i) = klic(16+i);
  TK3(i) = klic(24+i);
  TK4(i) = klic(32+i);
  TK5(i) = klic(40+i);
  TK6(i) = klic(48+i);
  TK7(i) = klic(56+i);
  TK8(i) = klic(64+i);
  TK9(i) = klic(72+i);
  TK10(i) = klic(80+i);
  TK11(i) = klic(88+i);
  TK12(i) = klic(96+i);
  TK13(i) = klic(104+i);
  TK14(i) = klic(112+i);
  TK15(i) = klic(120+i);         %nejvice dulezity bajt

end

for i = 1:1:16              %naplneni hlavnich promennych
    PK0(i) = IVh(i);
    PK1(i) = IVh(16+i);
end
PK2 = [TA0;TA1];
PK3 = [TA2;TA3];
PK4 = [TA4;TA5];

for  i = 0:1:3               %vlastni vypocet

      s = zeros;
      s2 = zeros;
      q = zeros;
```

```
r = logical(xor(PK4,[TK0;TK1]));
s = bi2de(r');
s2 = bi2de(PK0');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK0 = logical(de2bi(q,16));
r = logical(xor(PK0,[TK4;TK5]));
s = bi2de(r');
s2 = bi2de(PK1');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK1 = logical(de2bi(q,16));
r = logical(xor(PK1,[TK8;TK9]));
s = bi2de(r');
s2 = bi2de(PK2');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK2 = logical(de2bi(q,16));
r = logical(xor(PK2,[TK12;TK13]));
s = bi2de(r');
s2 = bi2de(PK3');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK3 = logical(de2bi(q,16));
r = logical(xor(PK3,[TK0;TK1]));
s = bi2de(r');
s2 = bi2de(PK4');
q = tabulka(s) +s2 +i-65536*fix((tabulka(s) +s2+i)/65536);
PK4 = logical(de2bi(q,16));
r = logical(xor(PK4,[TK2;TK3]));
s = bi2de(r');
s2 = bi2de(PK0');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK0 = logical(de2bi(q,16));
r = logical(xor(PK0,[TK6;TK7]));
s = bi2de(r');
s2 = bi2de(PK1');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK1 = logical(de2bi(q,16));
r = logical(xor(PK1,[TK10;TK11]));
s = bi2de(r');
s2 = bi2de(PK2');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
```

```
     PK2 = logical(de2bi(q,16));
     r = logical(xor(PK2,[TK14;TK15]));
     s = bi2de(r');
     s2 = bi2de(PK3');
     q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
     PK3 = logical(de2bi(q,16));
     r = logical(xor(PK3,[TK2;TK3]));
     s = bi2de(r');
     s2 = bi2de(PK4');
     q = tabulka(s) +s2 +2*i+1 -65536*fix((tabulka(s)
+s2+2*i+1)/65536);
     PK4 = logical(de2bi(q,16));

end

mixklic = [PK0;PK1;PK2;PK3;PK4];
```

B.4 Zdrojový kód fáze 2 mixování klíče

```
function [wepklic,TSC1d0]  = faze2(klic,mixklic,IVd,tabulka)
coder.extrinsic('vec2mat', 'bi2de','de2bi');
%vytvoreni pomocnych promennych
PK0=rand(16,1)<0.5;
PK1=rand(16,1)<0.5;
PK2=rand(16,1)<0.5;
PK3=rand(16,1)<0.5;
PK4=rand(16,1)<0.5;
PK5=rand(16,1)<0.5;
TSC=rand(16,1)<0.5;
TSC0=rand(8,1)<0.5;
TSC1=rand(8,1)<0.5;
TK0=rand(8,1)<0.5;
TK1=rand(8,1)<0.5;
TK2=rand(8,1)<0.5;
TK3=rand(8,1)<0.5;
TK4=rand(8,1)<0.5;
TK5=rand(8,1)<0.5;
TK6=rand(8,1)<0.5;
TK7=rand(8,1)<0.5;
TK8=rand(8,1)<0.5;
TK9=rand(8,1)<0.5;
TK10=rand(8,1)<0.5;
TK11=rand(8,1)<0.5;
TK12=rand(8,1)<0.5;
TK13=rand(8,1)<0.5;
TK14=rand(8,1)<0.5;
TK15=rand(8,1)<0.5;


for i = 1:1:16
    PK0(i) = mixklic(i);
    PK1(i) = mixklic(16+i);
    PK2(i) = mixklic(32+i);
    PK3(i) = mixklic(48+i);
    PK4(i) = mixklic(64+i);
    TSC(i)=IVd(i);
```

```
end
PK5 = logical(bsxfun(@plus,logical(PK4),logical(TSC)));
for i = 1:1:8                %naplneni pomocnych promennych
  TK0(i) = klic(i);    %nejmene dulezity bajt
  TK1(i) = klic(8+i);
  TK2(i) = klic(16+i);
  TK3(i) = klic(24+i);
  TK4(i) = klic(32+i);
  TK5(i) = klic(40+i);
  TK6(i) = klic(48+i);
  TK7(i) = klic(56+i);
  TK8(i) = klic(64+i);
  TK9(i) = klic(72+i);
  TK10(i) = klic(80+i);
  TK11(i) = klic(88+i);
  TK12(i) = klic(96+i);
  TK13(i) = klic(104+i);
  TK14(i) = klic(112+i);
  TK15(i) = klic(120+i);         %nejvice dulezity bajt
  TSC0(i) = IVd(i);
  TSC1(i) = IVd(i+8);
end
%provedeni substituci
     s = zeros;
      s2 = zeros;
      s3 = zeros;
      q = zeros;
     r = logical(xor(PK5,[TK0;TK1]));
     s = bi2de(r');
     s2 = bi2de(PK0');
     q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
     PK0 = logical(de2bi(q,16));
     r = logical(xor(PK0,[TK2;TK3]));
     s = bi2de(r');
     s2 = bi2de(PK1');
     q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
     PK1 = logical(de2bi(q,16));
     r = logical(xor(PK1,[TK4;TK5]));
     s = bi2de(r');
```

```
s2 = bi2de(PK2');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK2 = logical(de2bi(q,16));
r = logical(xor(PK2,[TK6;TK7]));
s = bi2de(r');
s2 = bi2de(PK3');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK3 = logical(de2bi(q,16));
r = logical(xor(PK3,[TK8;TK9]));
s = bi2de(r');
s2 = bi2de(PK4');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK4 = logical(de2bi(q,16));
r = logical(xor(PK4,[TK10;TK11]));
s = bi2de(r');
s2 = bi2de(PK5');
q = tabulka(s) +s2 -65536*fix((tabulka(s) +s2)/65536);
PK5 = logical(de2bi(q,16));
%provedeni rotaci
r = logical(xor(PK5,[TK12;TK13]));
s = circshift(r,1);
s2 = bi2de(PK0');
s3 = bi2de(s');
PK0 = logical(de2bi(s2+s3-65536*fix((s2+s3)/65536),16));

r = logical(xor(PK0,[TK14;TK15]));
s = circshift(r,1);
s2 = bi2de(PK1');
s3 = bi2de(s');
PK1 = logical(de2bi(s2+s3-65536*fix((s2+s3)/65536),16));
s = circshift(PK1,1);
s2 = bi2de(PK2');
s3 = bi2de(s');
PK2 = logical(de2bi(s2+s3-65536*fix((s2+s3)/65536),16));
s = circshift(PK2,1);
s2 = bi2de(PK3');
s3 = bi2de(s');
PK3 = logical(de2bi(s2+s3-65536*fix((s2+s3)/65536),16));
s = circshift(PK3,1);
```

```
      s2 = bi2de(PK4');
      s3 = bi2de(s');
      PK4 = logical(de2bi(s2+s3-65536*fix((s2+s3)/65536),16));
      s = circshift(PK4,1);
      s2 = bi2de(PK5');
      s3 = bi2de(s');
      PK5 = logical(de2bi(s2+s3-65536*fix((s2+s3)/65536),16));

      %vytvoreni vystupniho klice
      RC4K0=rand(8,1)<0.5;
      RC4K1=rand(8,1)<0.5;
      RC4K2=rand(8,1)<0.5;
      RC4K3=rand(8,1)<0.5;
      RC4K0 = TSC1;
      t = [0 0 0 0 0 1 0 0];
      u = [1 1 1 0 1 1 1 1];
      RC4K1 = (and(or(TSC1,t'),u'));
      RC4K2 = TSC0;
      r = logical(xor(PK5,logical(bitsrl(int32([TK0;TK1]),1))));
      for i = 1:1:8
      RC4K3(i) = r(i);
      end

wepklic = [RC4K0;RC4K1;RC4K2;RC4K3;PK0;PK1;PK2;PK3;PK4;PK5];
TSC1d0 = [RC4K0;RC4K1;RC4K2];
```

B.5 Zdrojový kód MIC bloku

```
function vystupMIC  = fcn(MICklic,data,MACv,MACp)
%zpracovani erroru pri zadani spatnych delek promennych
if ( rem(length(data),8)~= 0)
    error ('Vstupni data nejsou delitelna 8')
end
if ( length(MACv)~=48)
    error ('MAC adresa vysilace nema delku 48 bitu')
end
if ( length(MACp)~=48)
    error ('MAC adresa prijimace nema delku 48 bitu')
end
if (isempty(data))
    error ('Na vstupu nejsou žádná data')
end



l=rand(32,1)<0.5;
r=rand(32,1)<0.5;
m=rand(32,1)<0.5;
data = [MACp;MACv;data];
%doplnkovy byte s hodnotou 0x5a
lastbyte= logical([0 1 0 1 1 0 1 0]);
zerobyte= logical([0 0 0 0 0 0 0 0]);
coder.extrinsic('vec2mat', 'bi2de','de2bi');

for i = 1:1:32 %rozdeleni mic
 l(i) = MICklic(i);
 r(i) = MICklic(32+i);

end
data1 = [data;lastbyte'];
a=length(data1);
%doplneni do 32 bitovych slov
    switch rem(a/8,4)
        case 1
            data2 = [data1;zerobyte';zerobyte';zerobyte'];
        case 2
```

```
            data2 = [data1;zerobyte';zerobyte'];
        case 3
            data2 = [data1;zerobyte'];
        otherwise
            data2 = data1;

    end
    %doplneni o 4 nulove bajty
    data3  = int32([data2;zerobyte';zerobyte';zerobyte';zerobyte']);

    y1=zeros(length(data3)/32,32,'int32');
    y1=vec2mat(data3,32);
    pom=rand(32,1)<0.5;
    l2 = zeros;
    r2 = zeros;
    %padding dokoncen
%vlastni zabezpeceni
for i = 1:1:length(data3)/32
    for j = 1:1:32
        m(j) = y1(i,j);
    end
l = logical(xor(l,m));
 pom = circshift(l,-17);
r = logical(xor(r,pom));
l2 = bi2de(l');
r2 = bi2de(r');
l = logical(de2bi(l2+r2-(2^32)*fix((l2+r2)/(2^32)),32));
pom = circshift(l,-16);
r = logical(xor(r,pom));
l2 = bi2de(l');
r2 = bi2de(r');
l = logical(de2bi(l2+r2-(2^32)*fix((l2+r2)/(2^32)),32));
pom = circshift(l,-3);
r = logical(xor(r,pom));
l2 = bi2de(l');
r2 = bi2de(r');
l = logical(de2bi(l2+r2-(2^32)*fix((l2+r2)/(2^32)),32));
pom = circshift(l,2);
r = logical(xor(r,pom));
```

```
l2 = bi2de(l');
r2 = bi2de(r');
l = logical(de2bi(l2+r2-(2^32)*fix((l2+r2)/(2^32)),32));
end

 vystupMIC = [l;r];
```

C WPA2

C.1 Zdrojový kód WPA2 S-Box

```
function [sbox,inverznisbox]  = substitucnitabulka()
polynom = zeros(1,1,'uint32');
inverze = zeros(256,1);
sboxout = zeros(256,1);
invsbox = zeros(256,1);
inverznisbox = zeros(256,1);
polynom = bin2dec ('100011011'); %generacni polynom
inverze(1) = 0;
pomi = zeros(1,1,'uint32');
pomj = zeros(1,1,'uint32');
%provedeni inverzního násobení v galloisove poli GF(2^8)
for i = 1:1:255
    pomi = uint32(i);
    for j = 1:1:255
        pomj = uint32(j);
        ab = zeros(1,1,'uint32');
        for  bit = 1:1:8
            if bitget (pomi,bit)
                pomjposun = bitshift (pomj, bit - 1);
                ab = bitxor (ab, pomjposun);
            end
        end
        for bit = 16:-1:9
            if bitget (ab,bit)
                bitovyposun = bitshift (uint32(polynom), bit - 9);
                ab = bitxor (ab, bitovyposun);
            end
        end
        if ab == 1
         break
        end

    end
    inverze(i+1) = j;
end
```

```
%generacni polynomy pro sbox  Rijndael
modulopolynom = bin2dec ('100000001');
nasobicipolynom = bin2dec ('00011111');
scitacipolynom = bin2dec ('01100011');

%vytvoreni vlastniho sbox vektoru
for i = 1:1:256
ab = zeros(1,1,'uint32');
  for  bit = 1:1:8
    if bitget (uint32(inverze(i)),bit);
nasobicipolposun = bitshift(uint32(nasobicipolynom), bit - 1);
      ab = bitxor (ab, nasobicipolposun);
    end
  end
  for bit = 16:-1:9
    if bitget (ab,bit)
      bitovyposun = bitshift(uint32(modulopolynom), bit - 9);
      ab = bitxor (ab, bitovyposun);
    end
  end

    sboxout(i)= bitxor(ab,scitacipolynom);
end
% vytvoreni inverzniho sboxu
for i = 1:1:256
    invsbox(sboxout(i) + 1) = i - 1;
end
%poslani sboxu a invsboxu na vystup
sbox = sboxout;
inverznisbox = invsbox;
```

C.2 Zdrojový kód bloku rozšíření klíče

```
function rozsirenyklic   = keyexpansion(klic,sbox)
coder.extrinsic('bi2de','reshape','vec2mat');
klicd = zeros(16,1);
klicr = zeros(16,8);
klicr = vec2mat(double(klic),8);
klicd = bi2de(klicr);
rozsirenyklic =zeros(44,4);
rozsirenyklic(1:4,1:4) = (reshape(klicd,4,4))';

for i = 5:1:44
        temp = uint32(rozsirenyklic(i-1,:));
    % u 4teho provedeni dalsich operaci
    if mod (i,4) == 1
        % provedeni prohazeni poradi
        temp = temp([2 3 4 1]);
          %substituce s sbox
        for j=1:1:4
            temp(1,j)=sbox(temp(1,j));
        end
        %rcon tabulka pouzivana v galloisove poli
        c=uint32(1);
        in = (i-1)/4;
        while(in ~= 1)
b = bitand(c,128);
c = bitshift(c,-1);
            if(b == 128)
c=bitxor(c,27);
            end
                in=in-1;
        end

        temp = bitxor(temp,[c;0;0;0]');
    end

    rozsirenyklic(i,:) =
bitxor(uint32(rozsirenyklic(i-4,:)),uint32(temp));
end
```

C.3 Zdrojový kód bloku vektorPNgen

```
function vektorPN   = vektorPNgen(data,SA,PN)
%generovani PN vektoru jako nasobek 128 dle delky dat
global A;
flag = ([0 0 0 0 0 0 0 1])';
flag2 = ([0 0 0 0 0 0 0 0])';
ctr = zeros(16,1);
x = ceil(length(data)/128);
y = zeros(128,1);
z= zeros(x*128,1);
for i = 1:1:x
y = de2bi(PN,48);
ctr = de2bi(i-1,16);
pom = [flag;flag2;SA;y';ctr'];
z((i-1)*128+1:i*128)= pom;

end
vektorPN = z;
A = PN;
```

C.4 Zdrojový kód bloku AES

```
function PNsifrovano   = AES(rozsirenyklic,sbox,PNvektor,data)
coder.extrinsic('bi2de','reshape','vec2mat');
%doplneni dat na nasobek 128b
datablock= ceil(length(PNvektor)/128);
datadoplnena = [PNvektor;zeros(datablock*128-length(PNvektor),1)];
datad = zeros(16,1);
datar = zeros(16*datablock,8);
datar = vec2mat(double(datadoplnena),8);
sifrovanadatavystup =uint32(zeros(16*8*datablock,1));
sifrovanadata =uint32(zeros(4,4));
% cyklovani dle poctu 128b bloku dat
for blok = 1:1:datablock
datad = bi2de(datar(((blok-1)*16+1):(blok*16),:));
pomocna =uint32(zeros(16,1));
sifrovanadata(1:4,1:4) = (reshape(datad,4,4))';
```

```
% xor data s klicem
sifrovanadata = bitxor(sifrovanadata(1:4,1:4),
uint32(rozsirenyklic(1:4,1:4)));

for i = 1:1:9
    %sbox
    for j=1:1:4
        for k=1:1:4
       sifrovanadata(k,j)=sbox(sifrovanadata(k,j)+1);
        end
    end
    %rotace v radcich
    sifrovanadata(2,:) = sifrovanadata(2,[2 3 4 1]);
    sifrovanadata(3,:) = sifrovanadata(3,[3 4 1 2]);
    sifrovanadata(4,:) = sifrovanadata(4,[4 1 2 3]);

    %promichani sloupcu
    pmat = uint32([2 3 1 1;1 2 3 1;1 1 2 3;3 1 1 2]);
    modulopol = bin2dec ('100011011');
    sifrovanadata2 = uint32(zeros(4,4));


    % vypocet pres kazdy prvek 4x4 B matice promichani sloupcu
for pomi = 1:1:4
    for pomj = 1:1:4
        temp = uint32(0);
        for vektor = 1:1:4
            %provedeni  násobení v galoisove poli GF(2^8)
            kombinace = zeros(1,1,'uint32');
           for  bit = 1:1:8
                if bitget (pmat(pomj,vektor),bit)
                    pomjposun = bitshift (uint32(
sifrovanadata(vektor,pomi)), bit - 1);
                    kombinace = bitxor (kombinace, pomjposun);
               end
           end
            for bit = 16:-1:9
```

```
              if bitget (kombinace,bit)
                bitovyposun = bitshift (uint32(modulopol), bit - 9);
                kombinace = bitxor (kombinace, bitovyposun);
              end
            end
             temp = bitxor (temp,kombinace);
        end
        %vystupni prvky nahrazujici puvodni
        sifrovanadata2(pomj,pomi) = temp;
    end
end
sifrovanadata = sifrovanadata2;
%pouziti round key

roundkey = (rozsirenyklic((1:4)+4*i,:))';
sifrovanadata = bitxor(sifrovanadata(1:4,1:4),
uint32(roundkey(1:4,1:4)));
end
% posledni cyklus je jiny jak predchozi
for j=1:1:4
   for k=1:1:4
       sifrovanadata(k,j)=sbox(sifrovanadata(k,j)+1);
   end
end
sifrovanadata(2,:) = sifrovanadata(2,[2 3 4 1]);
sifrovanadata(3,:) = sifrovanadata(3,[3 4 1 2]);
sifrovanadata(4,:) = sifrovanadata(4,[4 1 2 3]);
roundkey = (rozsirenyklic(41:44,1:4))';
sifrovanadata = bitxor(sifrovanadata(1:4,1:4),
uint32(roundkey(1:4,1:4)));

%prevedeni sifrovanych dat na bity
pomocna = reshape(sifrovanadata,16,1);
pomocna2 = de2bi(pomocna,8);
sifrovanadatavystup(((blok-1)*128+1):(blok*128),1)
= reshape(pomocna2',16*8,1);
end

PNsifrovano = logical(sifrovanadatavystup(1:length(data)));
```

D OBSAH PŘILOŽENÉHO DVD

- xvojti05.pdf – vlastní práce ve formátu PDF
- složka WEP – obsahuje funkční model WEP
- složka WPA – obsahuje funkční model WPA
- složka WPA2 – obsahuje funkční model WEP2